

Learning to Use Selective Attention and Short-Term Memory in Sequential Tasks

Published: *From Animals to Animats,
Fourth International Conference
on Simulation of Adaptive Behavior, (SAB'96)
Cape Cod, Massachusetts. September, 1996*

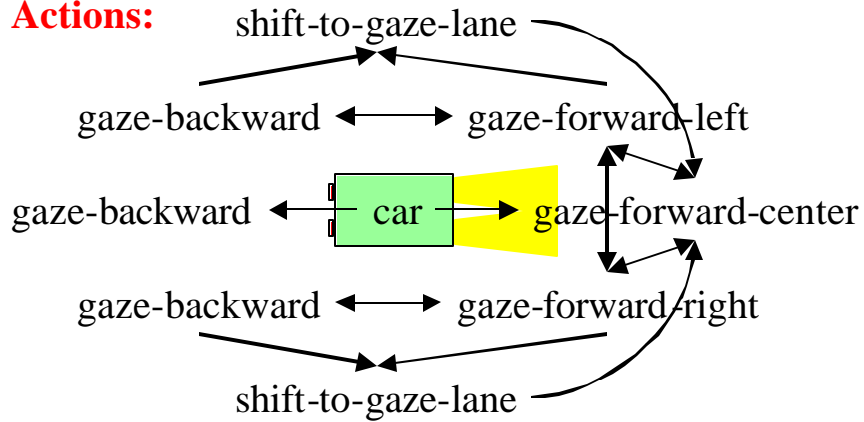
Written By:
Andrew Kachites McCallum

Presented By:
*James M. Vaccaro
CSE 254 June 4, 2002*

Scenario

Use Selective Attention & Action to Observe & React to Current State

Actions:

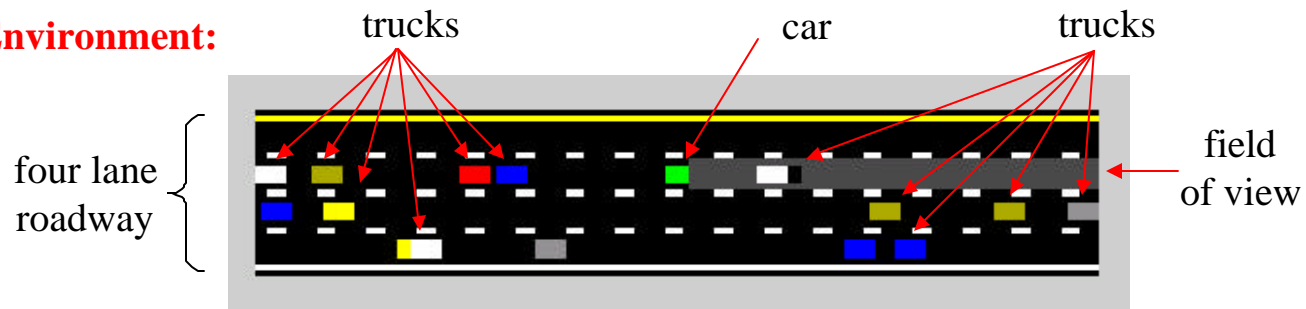


Sensors:

- Hear horn
- Gaze object
- Gaze side
- Gaze direction
- Gaze speed
- Gaze distance
- Gaze refined distance
- Gaze color

Application is a Sequential Task for Passing trucks in heavy traffic

Environment:



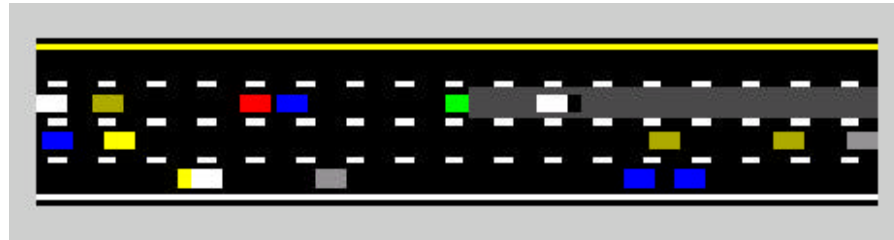
Rewards: -10 for scrapping a truck, -1 for causing a truck to beep, and 0.1 otherwise

Outline

- Issue: Dealing with *Too Much* and *Too Little* Information
- New Algorithm: U-Tree Algorithm
- U-Tree Algorithm Components
 1. Utile Distinction Memory
 2. Instance-Based Learning with Nearest Sequence Memory
 3. Utile Suffix Memory and Learning Tree
- U-Tree Algorithm Revisited
 1. Example of Combining Approaches
 2. Transition Instance Chain
 3. Learning U-Tree Algorithm
- Application & Demo
- Results, Inconsistencies & Summary

Dealing with *Too Much*

Too many possible sensory states, actions or observations



$|A| = 5$

$|D_1| = 2$ Hear horn

$|D_2| = 3$ Gaze object

$|D_3| = 3$ Gaze side

$|D_4| = 2$ Gaze direction

$|D_5| = 2$ Gaze speed

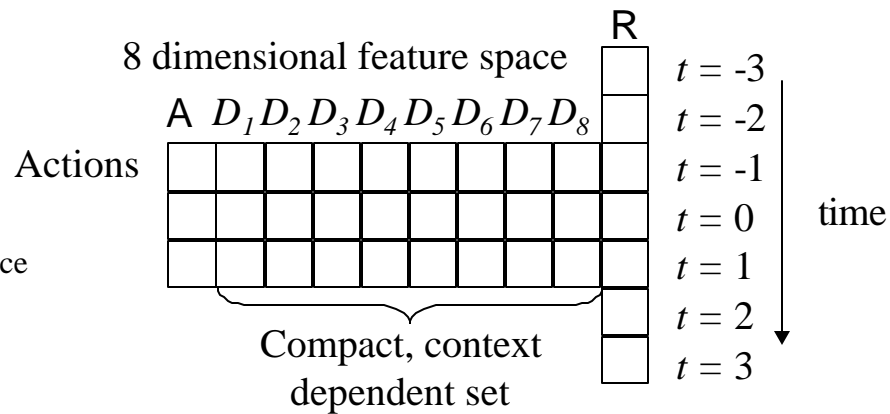
$|D_6| = 3$ Gaze distance

$|D_7| = 2$ Gaze refined distance

$|D_8| = 6$ Gaze color

$|R| = 3$

Rewards



and *Too Little* Information

Not all pertinent states can be observed at any one time

U-Tree Algorithm

Key feature is it simultaneously handles:

- “too much sensory data”
- “too little sensory data”

Two key structures:

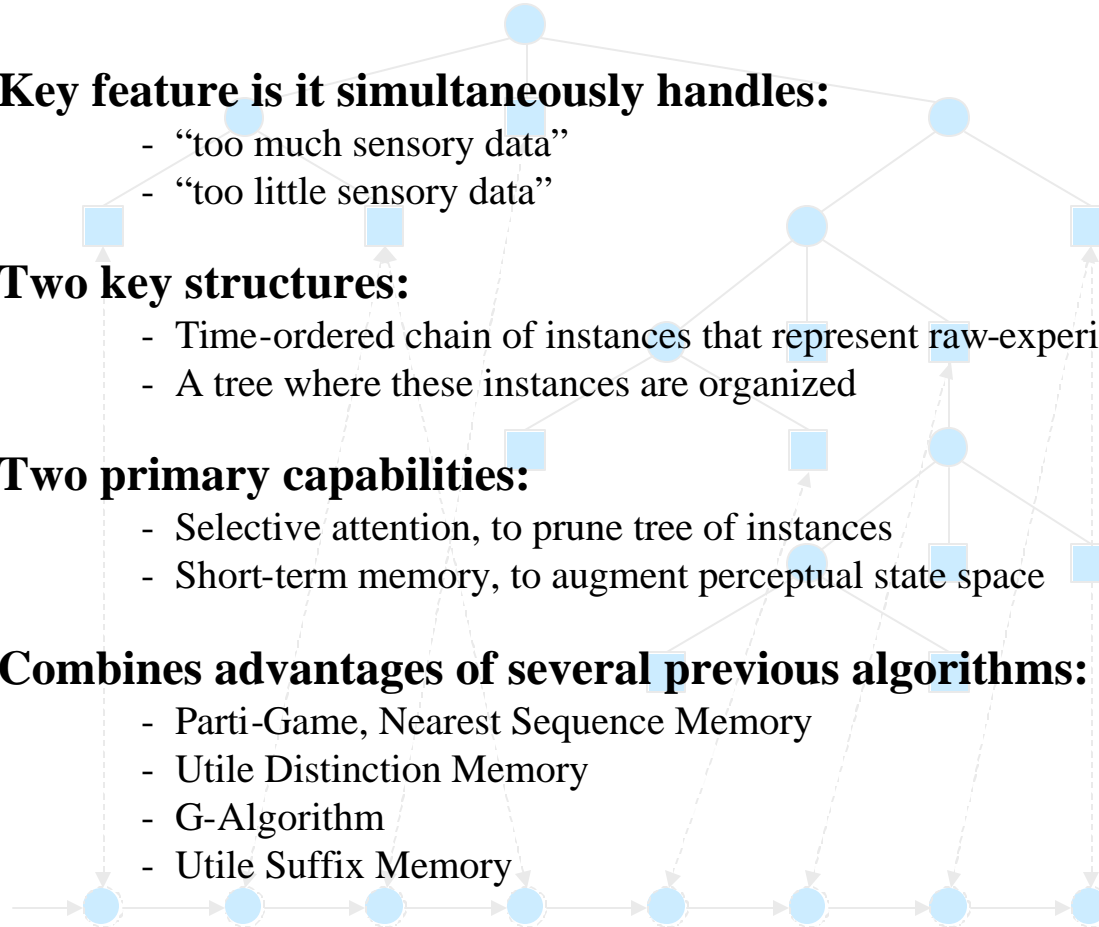
- Time-ordered chain of instances that represent raw-experience
- A tree where these instances are organized

Two primary capabilities:

- Selective attention, to prune tree of instances
- Short-term memory, to augment perceptual state space

Combines advantages of several previous algorithms:

- Parti-Game, Nearest Sequence Memory
- Utile Distinction Memory
- G-Algorithm
- Utile Suffix Memory



What Makes Up the U-Tree Algorithm?

(1) **Utile Distinction Memory [McCallum 1993]**

Uses a statistical technique, a utile distinction test to separate noise from task structure and keep only short-term memories that help predict reward

(2) Parti-Game [Moore, 1993] & **Nearest Sequence Memory [McCallum 1995]**

instance-based finding conjunctions among features

(3) G-Algorithm [Chapman 1989]

Agent can select which individual features to attend to

New Feature: Ability to divide percepts into components to perform selective attention

(1) & (2) **Utile Suffix Memory (USM) [McCallum 1995]**

Task: Three alternative methods make up the U-Tree algorithm

1st Technique: Utile Distinction Memory (UDM)

The utile distinction test distinguishes states that have different policy actions or different utilities, and merges states that have the same policy action and same utility.

Theorem: *The state distinctions necessary for representing the optimal policy are not necessarily sufficient for learning the optimal policy (proof by counter example)*

Notation:

State make-up: $s(t) = \langle a(t-1), o(t), r(t) \rangle$

States: $S = \{s_1, s_2, \dots, s_n\}$ where n is the number of states

Actions: $A = \{a_1, a_2, \dots, a_k\}$ where k is the number of possible actions

Observations: $O = \{o_1, o_2, \dots, o_m\}$ where m is the number of observed features

Observation probability: $P(o_j | s_j)$ Reward: r

State probability: $P(s_j | t)$ Utility or use of state: $U(s_j | t)$

Transition probability: $P(s_k | s_i, a_j)$ State-action value: $q(s_i, a_j)$

Utile Distinction Memory Algorithm

1. Agent's belief in s_j : $\hat{p}_j^t = \frac{1}{k} \sum_{i=1}^k \frac{o_{t,i}}{s_j} \mathbb{1}_{s_j/s_t = a_t}$

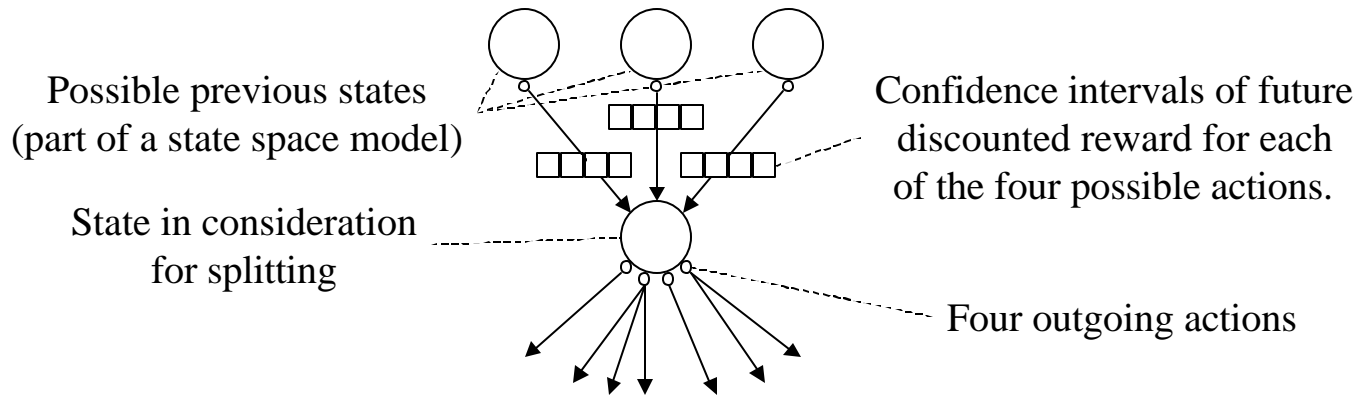
2. Current policy: $Q_{i,t}^{\rightarrow}, a_j = \sum_{i=1}^k \hat{p}_i^t q_{s_t, a_j}$

3. Update Q-values: $Q_{i,t}^{\rightarrow}(s_i, a_t) = \alpha \left(\beta p_i^t r_i + \sum_{j=1}^k Q_{i,t}^{\rightarrow}(s_j, a_t) \right)$
 α – learning rate
 β – temporal discount factor

4. Update expected utility: $U_{i,t}^{\rightarrow} = \max_a Q_{i,t}^{\rightarrow}(s_t, a)$

5. Choose best action: $a_{t+1} = \arg \max_a Q_{i,t}^{\rightarrow}(s_t, a)$

Evaluating Potential Node Splitting Possibilities



Calculate return values over time:

$\text{return}[m] = r[m]$
 for $t = m - 1$ to 0
 $\text{return}[t] = r[t] + ? * \text{return}[t + 1]$

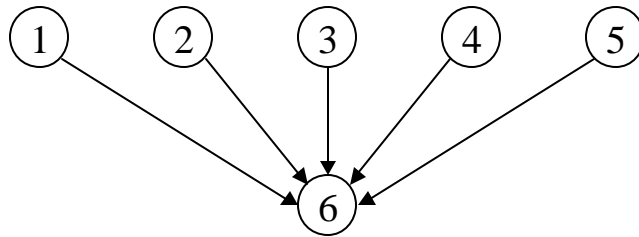
? is learning rate

Calculate upper/lower bound statistics:

for $t = 1$ to $m - 1$
 for all transitions, **trans**,
 that use action $A[t - 1]$
 $? = ? * ?_{\text{trans}}[t - 1]$
 $\text{trans.count}_{A[t]} += ?$
 $\text{trans.sum}_{A[t]} += ? * \text{return}[t]$
 $\text{trans.sumsquares}_{A[t]} += ? * (\text{return}[t])^2$

Splitting Nodes Based on Confidence Intervals

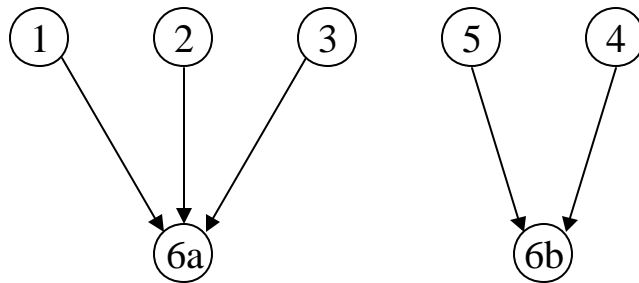
Before split:



Confidence intervals on future discounted reward:

r_1	
r_2	
r_3	
r_4	
r_5	

After split:



Node Splitting Test: Kolmogorov-Smirnov (KS)

Statistical tests (**in general**, for KS *and with details*):

1. null hypothesis

KS: Calculate or use Relative frequency distribution $F_0(X)$

Assume $F_0(X)$ is a student t function with $n-1$ degrees of freedom.

Calculate a sample mean and standard deviation from observed future discounted rewards for each action.

2. an alternative hypothesis

KS: Observed cumulative frequency distribution $S_n(X)$

Store future discounted reward distributions for each action and set it equal to $S_n(X)$.

3. decision maker

KS: $D = \max(|F_0(X) - S_n(X)|)$

Calculate the largest absolute difference between $F_0(X)$ and $S_n(X)$.

4. rejection region

KS: For specified α and sample size n reject if D exceeds $a = ?$

Promote split nodes from the previous state if rejected.

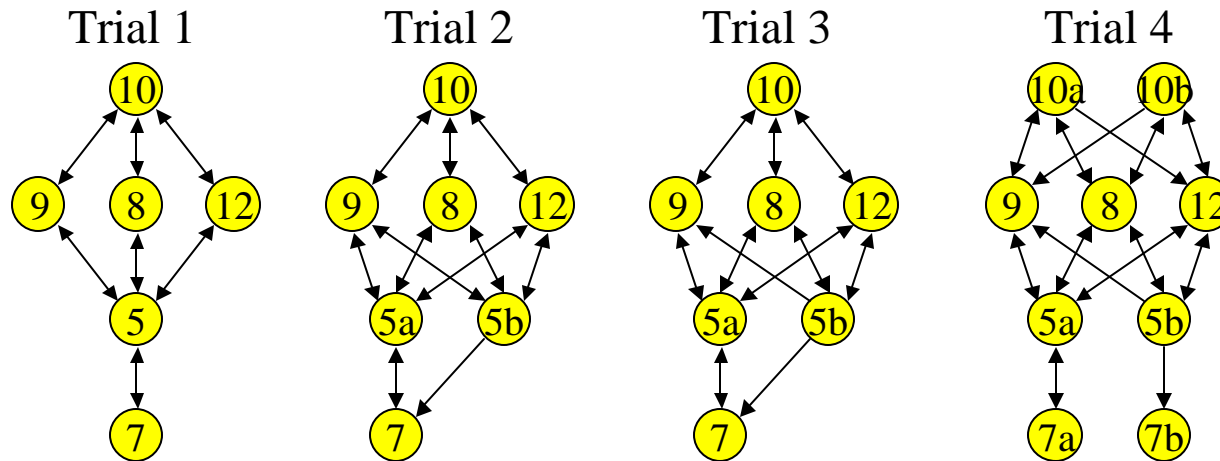
Utile Distinction Memory (UDM) Example

Grid-world:

	9	10a	8	10b	12	
	5a		5b		5a	
	7a		7b		7a	

Theorem: *The state distinctions necessary for representing the optimal policy are not necessarily sufficient for learning the optimal policy*

Note: Proof by counter example.

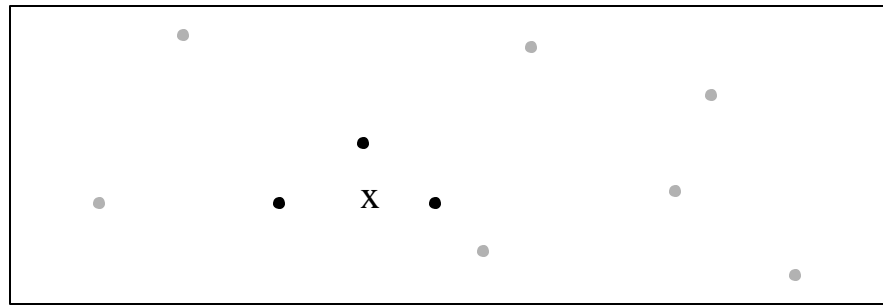


2nd Technique: Nearest Sequence Memory (NSM)

Learning in a Geometric Space

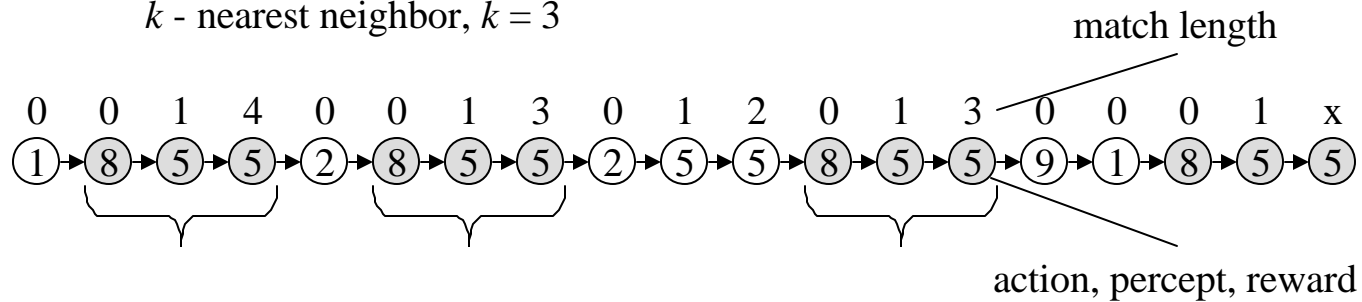
k - nearest neighbor, $k = 3$

2 dimensions



Learning in a Sequence Space

k - nearest neighbor, $k = 3$



Nearest Sequence Memory Algorithm (Step 1 & 2)

Step 1:

Find k -nearest neighbors of chained events:

$$n(s_i, s_j) = \begin{cases} 1 + n(s_{i-1}, s_{j-1}), & \text{if } (a_{i-1} = a_{j-1}) \wedge (o_{i-1} = o_{j-1}) \wedge (r_{i-1} = r_{j-1}) \\ 0, & \text{otherwise} \end{cases}$$

Vote for the best neighbors and resolve ties with the nearest:

$$v(s_i) = \begin{cases} 1, & \text{if } n(s_i, s_j) \text{ is among the } k \text{ max } n(s_i, s_j) \\ 0, & \text{otherwise} \end{cases}$$

Step 2:

Average over k -voting states to determine the Q-values:

$$Q_t(a_i) = \sum_{s_j/a_j/a_i} v(s_j) / k$$

Nearest Sequence Memory Algorithm (Step 3, 4 & 5)

Step 3:

Choose best action: $a_{t+1} = \arg \max_a Q_t(a)$

Step 4:

Increment the time counter: $t = t + 1$.

Create s_t ; record in it: a_t, o_t, r_t

Step 5:

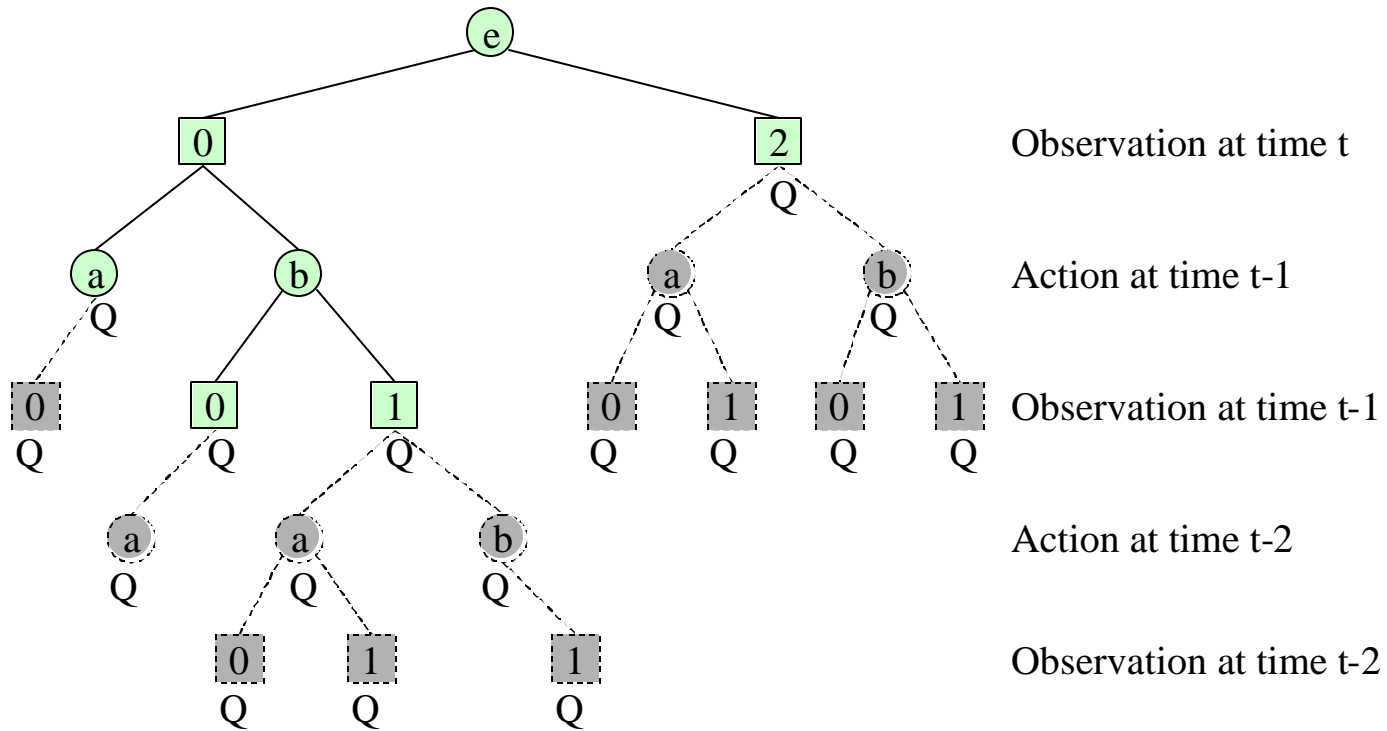
Update voted Q-values: $Q_t(s_i, a_i) = \alpha v(s_i, a_i) + (1 - \alpha) Q_{t-1}(s_i, a_i)$
 α – learning rate
 β – temporal discount factor

Update Expected Utility: $U_t = \max_a Q_t(a)$

Note: Utile Distinction Memory (UDM) uses value $q(s_t, a_t)$ while Nearest Sequence Memory (NSM) uses value $q(s_t)$ and UDM uses prob. $p_i(t)$ while NSM uses vote $v(s_t)$.

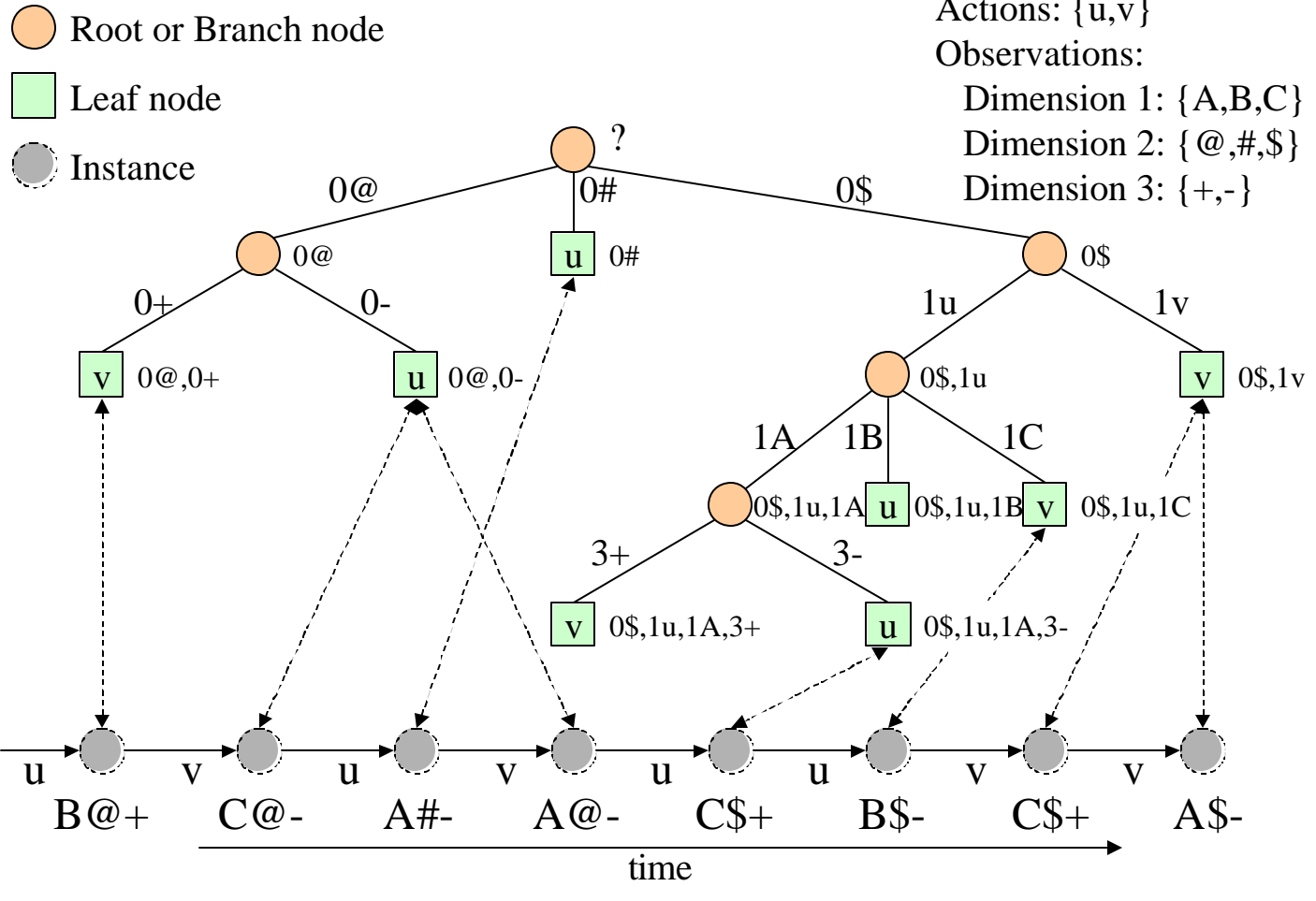
3rd Technique: Utile Suffix Memory (USM)

Tree initially branches for every dimension of the feature space.



Percepts are integer, actions are letters, fringe is in dashes and nodes labeled with Q are nodes that hold Q-values.

New Technique: U-Tree Revisited with an Example



U-Tree Algorithm Notation

Actions

Finite Set $A = \{a_1, a_2, \dots, a_{|A|}\}$ where $a_t \in A$

Rewards

Scalar Range $\mathbb{R} \ni x, y$ where x, y are reals $r_{t+1} \in \mathbb{R}$

Observations

Finite Set $O = \{o_1, o_2, \dots, o_{|O|}\}$ where $o_{t+1} \in O$

U-Tree Algorithm Observations in Further Detail

m perceptual features $\mathbf{D} = \{D_1, D_2, \dots, D_m\}$

Each feature is a finite set $D_d = \{D_{d,1}, D_{d,2}, \dots, D_{d,|D_d|}\}$

For each dimension at time t $\rightarrow \mathbf{d}_t$

Set of observations $o_t = \langle o^1_t, o^2_t, \dots, o^m_t \rangle$

The size of $\Theta = \prod_{d=1}^m |D_d|$

$|A| = 5$

$|D_1| = 2$ Hear horn

$|D_2| = 3$ Gaze object

$|D_3| = 3$ Gaze side

$|D_4| = 2$ Gaze direction

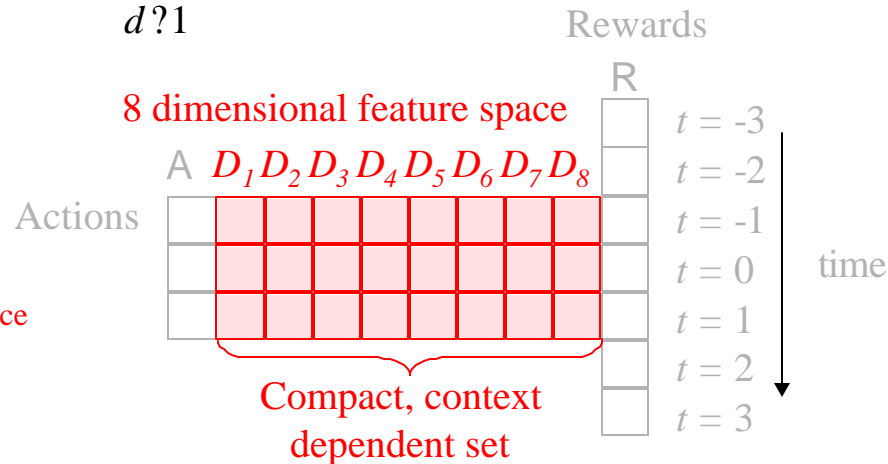
$|D_5| = 2$ Gaze speed

$|D_6| = 3$ Gaze distance

$|D_7| = 2$ Gaze refined distance

$|D_8| = 6$ Gaze color

$|R| = 3$

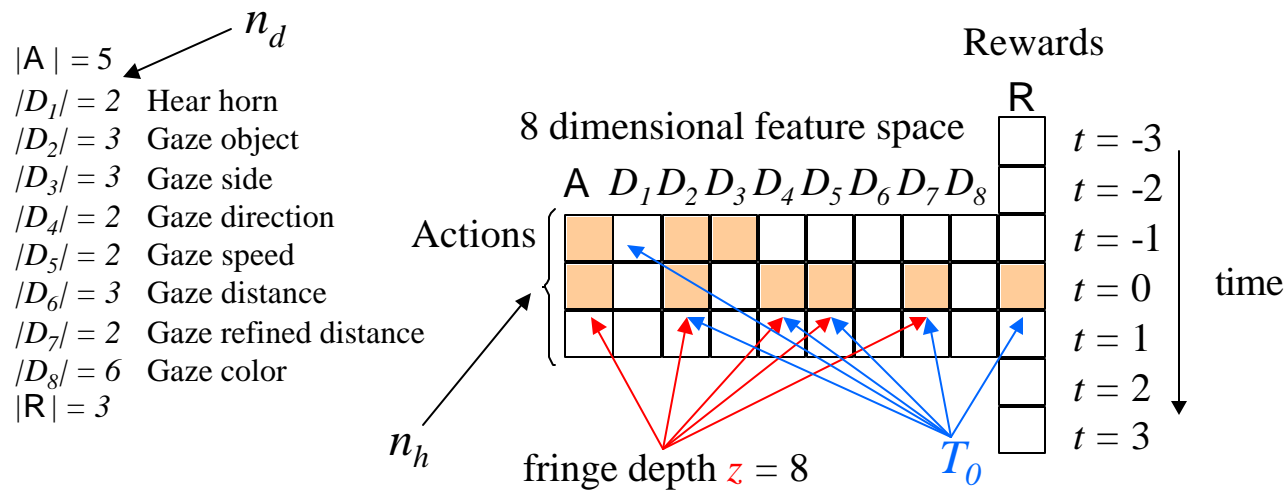


U-Tree Algorithm: Transition Instance

Transition Instance is 4 dimensional: $T_t \ ? \ \langle T_{t?1}, a_{t?1}, o_t, r_t \rangle$

where $T_{i?1}$ precedes T_i in chain.

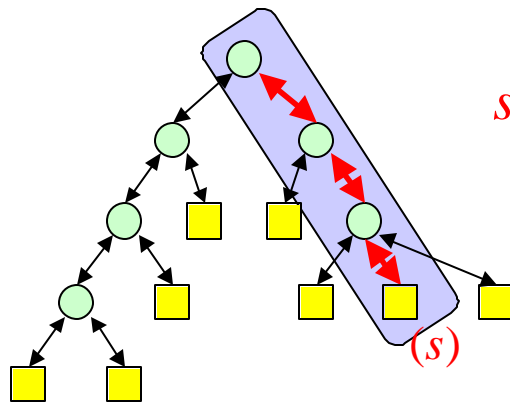
Chain of instances is linked by time and perception: $n_h \ n_d$



Each perceptual feature D_1, D_2, \dots, D_m has $|D_{n_d}|$ children.

U-Tree Algorithm: Transition Instance Tree

s conjunction: combined set of labels which form a composite state



$$s_t = \langle a_i, a_{i+1}, \dots, a_{i+ap}, O_j, O_{j+1}, \dots, O_{j+op}, r_t \rangle$$

Where i is a timestamp and ap is the number of actions used and where j represents a dimension and timestamp and op is the number of observations used

The union of all utility instances is $T(s)$ or T_s ? $T_s \rightsquigarrow T_t$

Tree leaf is the function $L(T)$ which is state s' given state s and action a

As one traverses down the tree time goes backward

Q-value or policy of taking next step in state conjunctions is $Q(s,a)$

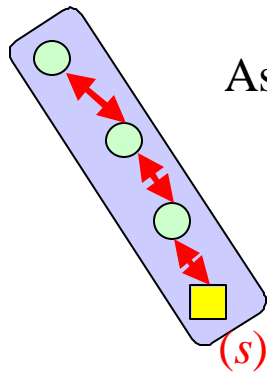
U-Tree Algorithm: Steps 1, 2 & 3

Step 1: Start with an empty root node with no distinctions \odot

Step 2: Choose a_{t+1} such that, $a_{t+1} \stackrel{?}{=} \operatorname{argmax}_{a \in A} Q \cdot \mathcal{T}_t \cdot a$
 Alternatively, with probability ϵ , the agent explores by choosing a random action instead.

Step 3: Agent makes move in environment

Records transition: $\odot \mathcal{T}_t \stackrel{?}{=} \langle \mathcal{T}_{t+1}, a_{t+1}, o_t, r_t \rangle$



As a chain of instances:



Continue down until reach leaf node s

U-Tree Algorithm: Step 4

For each time step taken, include one step of value iteration using dynamic programming [Bellman 1957]

$$Q_{s,a} \leftarrow R_{s,a} + \Pr_{s'/s,a} U_{s'}$$

where:

$$U_s = \max_{a \in A} Q_{s',a} \quad \text{utility of state } s'$$

(Distinctions that help predict reward)

$$R_{s,a} = \frac{\sum_{T_i \in T_{s,a}} r_i}{|T_{s,a}|} \quad \text{immediate reward}$$

$$\Pr_{s'/s,a} = \frac{|T_i \in T_{s,a} \text{ s.t. } L(T_i) = s'|}{|T_{s,a}|} \quad \text{estimated probability}$$

$T(s,a)$ set of all instances in the node s which also executed action a

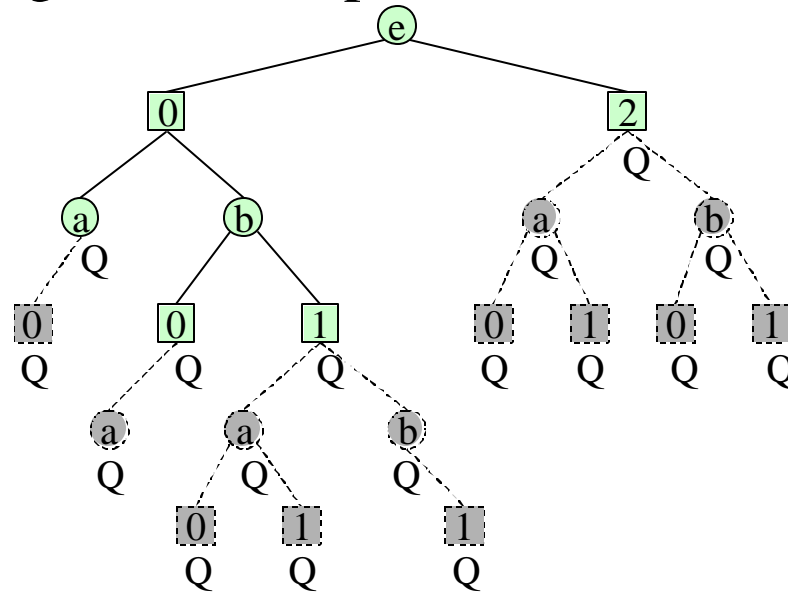
U-Tree Algorithm: Step 5

After every k time steps,
perform Kolmogorov-Smirnov test
to compare the distributions
of future discounted rewards
(as described earlier)

$$\Pr(L(T_{i+1})) = \Pr(s'|s, a)$$

Expected Future Discounted
Reward for Instance T_i :

$$\mathbb{E}_{L(T_i)} [r_i + \gamma \mathbb{E}_{L(T_{i+1})} [r_{i+1} + \gamma \mathbb{E}_{L(T_{i+2})} [r_{i+2} + \dots]]]$$



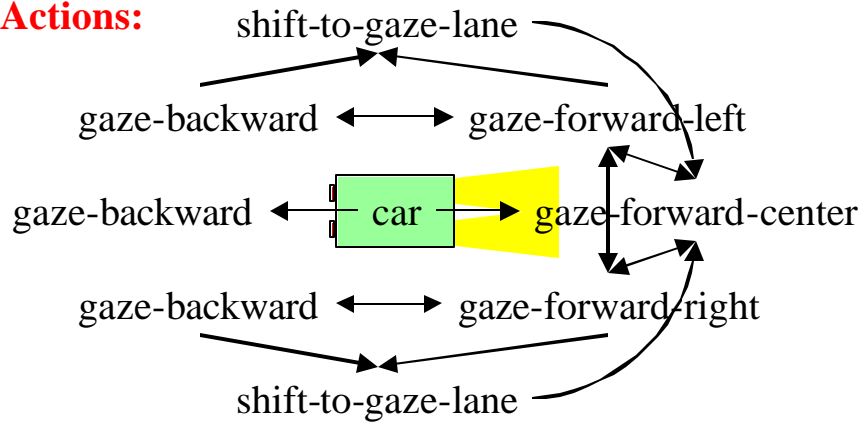
The number of all possible permutations of observations and actions to a fixed
depth z , using maximum history index h is $(|D| + 1)^z$

Increment t and return to Step 2.

Application

Use Selective Attention & Action to Observe & React to Current State

Actions:

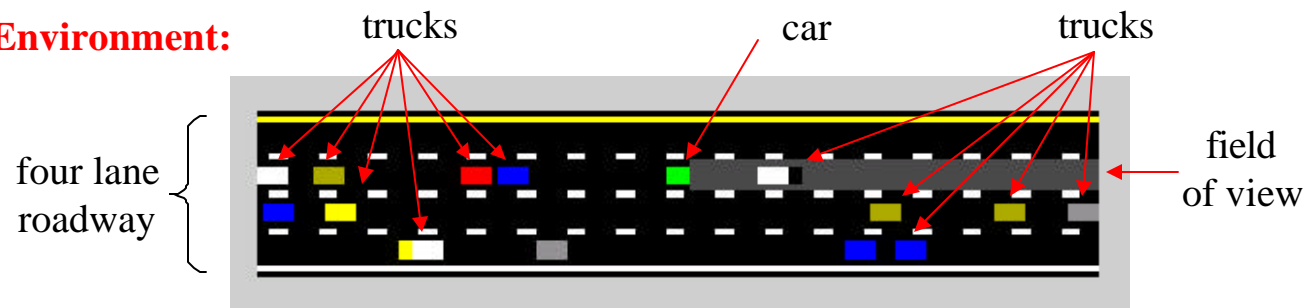


Sensors:

- Hear horn
- Gaze object
- Gaze side
- Gaze direction
- Gaze speed
- Gaze distance
- Gaze refined distance
- Gaze color

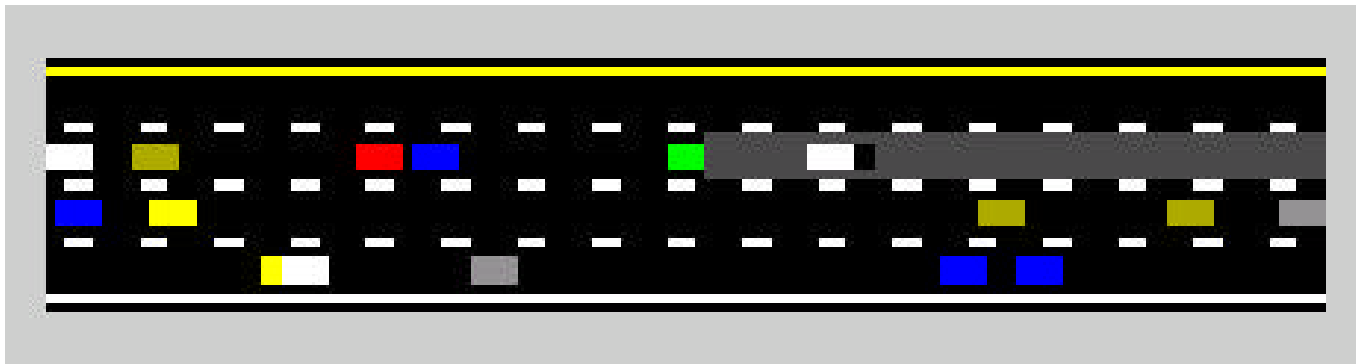
Application is a Sequential Task for Passing trucks in heavy traffic

Environment:



Rewards: -10 for scrapping a truck, -1 for causing a truck to beep, and 0.1 otherwise

The Application Environment



Agent's Car (green)

Speed: 16 m/s

66 m visual horizon
front and back and
one lane either side

Slow Trucks (any color)

Speed: 12 m/s

Continue speed
and lane in
every case

Fast Trucks (any color)

Speed: 20 m/s

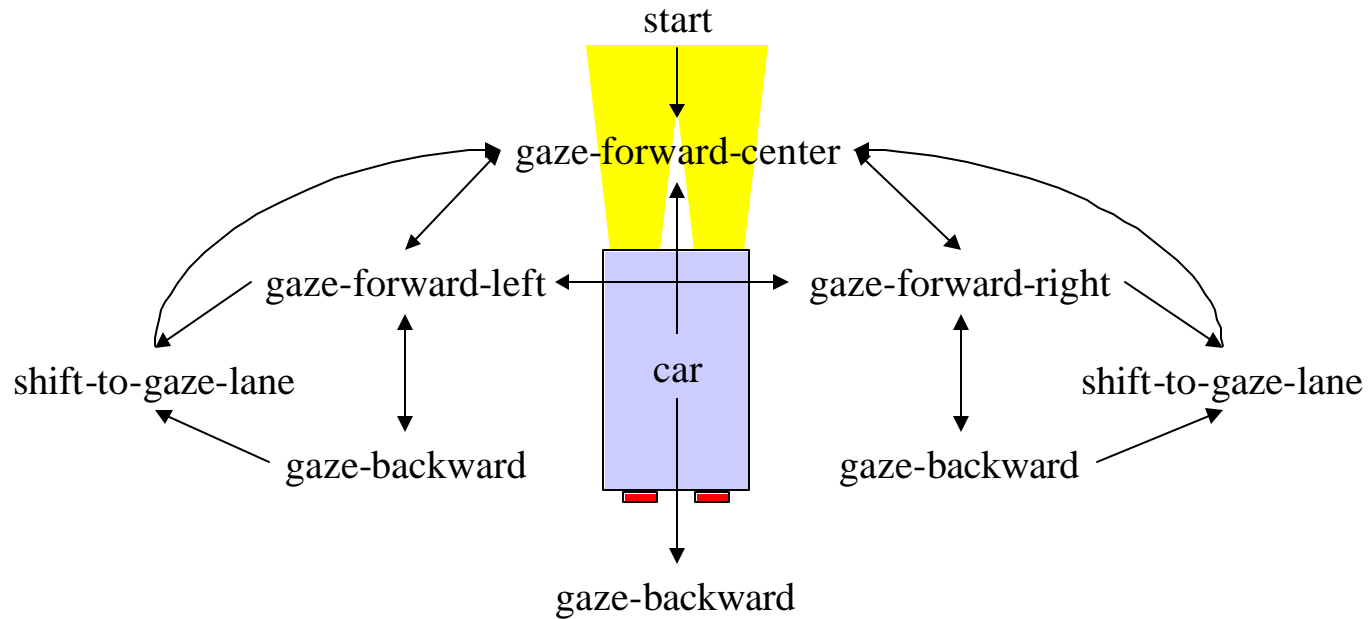
Slow down when
meeting other trucks or car,
and beeps at car in this case

New truck probability: 0.5, equally slow or fast

Time Step: 0.5 seconds

Rewards: -10 for scrapping a truck, -1 for causing a truck to beep, and 0.1 otherwise

Visual Routines and Deictic Actions



Five different actions, with context-dependent meaning

[Ulman 1984; Agre and Chapman 1987; Ballard *et al.*, 1996]

Agent Driver's Actions and Sensory System

Action	Description
Gaze-forward-left	Look at closest car in lane to the left.
Gaze-forward-center	Look at closest car in same lane as agent.
Gaze-forward-right	Look at closest car in lane to the right.
Gaze-backward	Look backwards at closest car in current gaze lane.
Shift-to-gaze-lane	Steer car into lane where agent is looking.

Dimension	Size = n_d	Values
Hear horn	2	yes, no
Gaze object	3	truck, shoulder, road
Gaze side	3	left, center, right
Gaze direction	2	forward, backward
Gaze speed	2	looming, receding
Gaze distance	3	far, near, nose
Gaze refined distance	2	far-half, near-half
Gaze color	6	red, blue, yellow, white, gray, tan

DEMO

The Application Parameter Values

U-Tree was trained for 10,000 steps

For Kolmogorov-Smirnov test, probability of $p = 0.0001$ was used

$k = 1000$, for time between splitting fringe nodes

First 2,000 steps exploration probability was 1.0

Next 4,000 steps exploration probability was 0.4

Next 2,000 steps exploration probability was 0.2

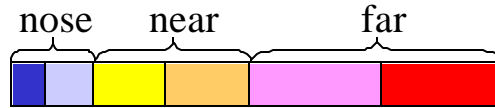
Last 2,000 steps exploration probability was 0.1

Results: Selected Nodes From 51 Leaves

- $t=0$ perception dimension [Gaze object = truck]
- $t=0$ perception dimension [Gaze side = center]
- $t=0$ perception dimension [Gaze distance = nose]

Reward at time t : -10

Gaze refined distance:

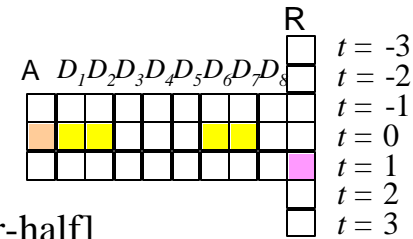
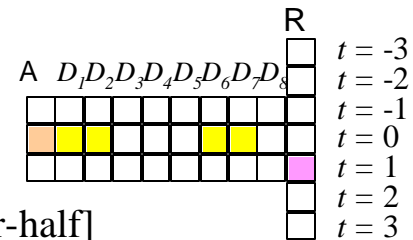
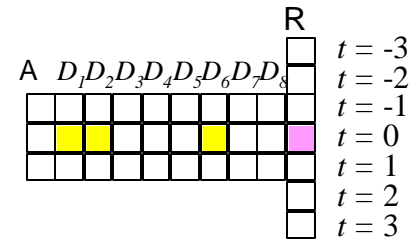


- $t=0$ perception dimension [Gaze object = truck]
- $t=0$ perception dimension [Gaze side = center]
- $t=0$ perception dimension [Gaze distance = near]
- $t=0$ perception dimension [Gaze refined distance = far-half]

Choosing Action: Gaze Center

- $t=0$ perception dimension [Gaze object = truck]
- $t=0$ perception dimension [Gaze side = left]
- $t=0$ perception dimension [Gaze distance = far]
- $t=0$ perception dimension [Gaze refined distance = far-half]

Choosing Action: Go to Gaze-Lane



Inconsistencies or *Possible Improvements*

- Tree is ranked and all antecedents must start from same point (e.g., seems as though trains slower and faster trucks separately).
- Cannot avoid the curse of dimensionality on the fringe nodes (i.e., $(h(|D|+1))^z$).
- $p = 0.0001$ for Kolmogorov-Smirnov test does not filter noise as predicted (needs more experiments)
- How the distribution function is formulated is unknown, must assume the same as used in Utile Distinction Memory (needs more experiments)
- There are no specifications about agent's ability to handle shoulder (minor detail)
- Dimension of vehicles is never specified (minor detail)

Discussion of Results

- U-Tree successfully combines selective attention, short-term memory, and instance-based reinforcement learning for a sequential task.
- During 5,000 step test run after learning, the policy committed only 67 collisions, giving a 32 % improvement from a hand-coded policy and a 91% improvement from a random policy.
- Learned policy has 51 leaves and irrelevant sensor attributes such as color were never used in these states.
- Number of leaf nodes is significantly fewer than the number of possible rules.
- Handles hidden and avoids useless states well.

Summary

