The Brainstormers -
Learning on the Tactical Level

**GOAL:** learn to use the basic skills by success or failure

**Tactics**
- actions: basic skill with varying length (kick, intercept, go2pos, ...)
- need for cooperation ⇒ multi-agent reinforcement learning
- clearly defined goal: score!!

**AI used in competition team**
- estimation of success probabilities (used in 1999; (Boek, Riedmiller, 2000))
- Depth-limited search for with-ball tactical decisions (since 2000)
- Neural network based cooperative attack positioning (since 2001)
- Neural network based complete cooperative attack behaviour (since 2002)

**Realized, but in research stage**
- Feature-Q-learning based attack (double-passes! (Ehrlmann, Riedmiller))

**Research Issues**
- Reinforcement learning in cooperative multi-agent system
- MAS-RL with features
- learning to communicate (Moore, Schneider, Riedmiller, 2001)
- incorporation of prior knowledge (constraints, special actions (homepositions), partially fixed policies (goal kick))
- theory: MAS-q algorithm for deterministic domains (Lauer, Riedmiller, 2000)
- theory: MAS-q algorithm for stochastic domains (Lauer, Riedmiller, 2001)
- theory: MAS-RL under varying conditions (e.g. exchange of information during learning): NAI, PAM, PSI, ...

**Learning 7 vs 8 Attack**
- individual decisions;
- actions: no-ball: 8 directions, default position
  - with-ball: pass, dribble, hold-ball
  - prior knowledge: intercept and goalshot are decided otherwise with priority
  - cooperate (central) learning and distribution of value function
  - => theoretically justified: optimal cooperative policy can be found
  - practical compromises: model based instead of Q-learning
  - use of an approximate model
  - raw coding of positions: (7 + 8) × 2 × 4 = 34 inputs

**MAS learning algorithm**

```java
// For all Situations s repeat n times
for (s in situations) {
    play_and_record(s);
} /* until sequence terminates */

// evaluate&generate_training_patterns();
// train_NN_for_Rp(s);
// distribute_NN();
// Repeat playing
```

**Results - example: positioning**

<table>
<thead>
<tr>
<th>Policy type</th>
<th>Average Seq.Length</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>random value function</td>
<td>139.5</td>
<td>0.16 (0.10)</td>
</tr>
<tr>
<td>attack 2000 (hardcoded)</td>
<td>129.4</td>
<td>0.18 (0.12)</td>
</tr>
<tr>
<td>Neuro Positioning (2001)</td>
<td>118.0</td>
<td>0.23 (0.18)</td>
</tr>
</tbody>
</table>

improved by 1.7 over viceworld champion team 2000!

**Individual action selection**

```java
GenerateActionSet();
for (all actions) {
    AbstractModel(s, s');
    evaluate(s')
    determine_best_action(s');
}

GenerateActionSet(): depends on situation
(w. ball, no. ball),
estimation of success
Abstract Model(): Rough estimation of successor state
```

---

Adaptive Systems Group
Prof. Dr. Martin Riedmiller, Artur Merke
Lehrstuhl Informatik 1
University of Dortmund, Germany

sponsored by DFG under grant SPP 1125