1. **Options**: Use “exit this room” high-level action

2. **Hierarchical Reinforcement Learning**: Attack, Mine Gold, Explore subtasks

3. **Learning from Demonstration**: Control Mario and have him learn to mimic you

4. **Ensembles**: Combine multiple classifiers to improve accuracy

5. **Unsupervised Learning**: Which players have similar playing styles?

6. **Active Learning**: Program asks which points should be labeled

7. **Other?**
• Monday: Project idea
• What to cover next?
Combination Techniques

1. \[ R'(s, a) = R(s, a) + (\text{weight} \times \hat{H}(s, a)). \]

2. \[ \overrightarrow{f'} = \overrightarrow{f} \cdot \text{append}(\hat{H}(s, a)). \]

3. Initially train \( Q(s, a) \) to approximate \( (\text{constant} \times \hat{H}(s, a)). \)

4. \[ Q'(s, a) = Q(s, a) + \text{constant} \times \hat{H}(s, a). \]

5. \[ A' = A \cup \text{argmax}_a[\hat{H}(s, a)]. \]

6. \[ a = \text{argmax}_a[Q(s, a) + \text{weight} \times \hat{H}(s, a)]. \]

7. \[ P(a = \text{argmax}_a[\hat{H}(s, a)]) = p. \text{ Otherwise original RL agent’s action selection mechanism is used.} \]

8. \[ R'(s_t, a) = R(s, a) + \text{constant} \times (\phi(s_t) - \phi(s_{t-1})), \text{ where } \phi(s) = \max_a H(s, a). \]
Combination Techniques

1. \( R'(s, a) = R(s, a) + (\text{weight} \times \hat{H}(s, a)) \).

2. \( \overrightarrow{f'} = \overrightarrow{f} \cdot \text{append}(\hat{H}(s, a)) \).

3. Initially train \( Q(s, a) \) to approximate \( (\text{constant} \times \hat{H}(s, a)) \).

4. \( Q'(s, a) = Q(s, a) + \text{constant} \times \hat{H}(s, a) \).

5. \( A' = A \cup \arg \max_a [\hat{H}(s, a)] \).

6. \( a = \arg \max_a [Q(s, a) + \text{weight} \times \hat{H}(s, a)] \).

7. \( P(a = \arg \max_a [\hat{H}(s, a)]) = p. \) Otherwise original RL agent’s action selection mechanism is used.

8. \( R'(s_t, a) = R(s, a) + \text{constant} \times (\phi(s_t) - \phi(s_{t-1})) \), where \( \phi(s) = \max_a H(s, a) \).
Combination Techniques

1. \( R'(s, a) = R(s, a) + (\text{weight} \times \hat{H}(s, a)) \).

2. \( \overrightarrow{v} = \overrightarrow{f}.\text{append} (\hat{H}(s, a)) \).

3. Initially \( \pi(s, a) \) is approximated \( \text{constant} \times \hat{H}(s, a) \).

4. \( Q'(s, a) = Q(s, a) + \text{constant} \times \hat{H}(s, a) \).

5. \( A' = A \cup \text{argmax}_a [\hat{H}(s, a)] \).

6. \( a = \text{argmax}_a [Q(s, a) + \text{weight} \times \hat{H}(s, a)] \).

7. \( P(a = \text{argmax}_a [\hat{H}(s, a)]) = p. \) Otherwise original RL agent’s action selection mechanism is used.

8. \( R'(s_t, a) = R(s, a) + \text{constant} \times (\phi(s_t) - \phi(s_{t-1})), \) where \( \phi(s) = \max_a H(s, a) \).
Combination Techniques

1. \( R'(s, a) = R(s, a) + (\text{weight} \times \hat{H}(s, a)) \).
2. \( \vec{f}' = \vec{f}.\text{append}(\hat{H}(s, a)) \).
3. Initially train \( Q(s, a) \) to approximate \( (\text{constant} \times \hat{H}(s, a)) \).
4. \( Q'(s, a) = Q(s, a) + \text{constant} \times \hat{H}(s, a) \).
5. \( A' = A \cup \arg\max_a[\hat{H}(s, a)] \).
6. \( a = \arg\max_a[Q(s, a) + \text{weight} \times \hat{H}(s, a)] \).
7. \( P(a = \arg\max_a[\hat{H}(s, a)]) = p. \) Otherwise original RL agent’s action selection mechanism is used.
8. \( R'(s_t, a) = R(s, a) + \text{constant} \times (\phi(s_t) - \phi(s_{t-1})) \), where \( \phi(s) = \max_a H(s, a) \).
Combination Techniques

1. \[ R'(s, a) = R(s, a) + (\text{weight} \times \hat{H}(s, a)). \]
2. \[ \vec{f}' = \vec{f} \cdot \text{append}(\hat{H}(s, a)). \]
3. Initially train \( Q(s, a) \) to approximate \( (\text{constant} \times \hat{H}(s, a)) \).
4. \[ Q'(s, a) = Q(s, a) + \text{constant} \cdot \hat{H}(s, a) \]
5. \[ A' = A \cup \text{argmax}_a[\hat{H}(s, a)]. \]
6. \[ a = \text{argmax}_a[Q(s, a) + \text{weight} \times \hat{H}(s, a)]. \]
7. \[ P(a = \text{argmax}_a[\hat{H}(s, a)]) = p. \] Otherwise original RL agent’s action selection mechanism is used.
8. \[ R'(s_t, a) = R(s, a) + \text{constant} \times (\phi(s_t) - \phi(s_{t-1})), \]
where \( \phi(s) = \max_a H(s, a). \)
Experiments

• **domain**: Mountain Car
• **RL algorithm**: Sarsa(\(\lambda\))
  – **features**: a grid of 2D Gaussian RBFs over state; one grid for each action
  – **representation of Q**: linear model
  – **initialization of Q**: both opt. and pess.
  – **updates**: gradient descent
• 30 runs of 500 episodes
Experiments

Two predictive models used (from among 19 trainers):

\( \hat{H}_1 \): middling performance (9\textsuperscript{th})

\( \hat{H}_2 \): best performance
Defining success

Outperforming:

On the metrics:
- cumulative reward
- final performance

On both $\hat{H}_1$ and $\hat{H}_2$
(for comparison, TAMER-only mean performance is -109.1)
Almost complete successes

\[ R'(s, a) = R(s, a) + (weight \times \hat{H}(s, a)) \]

and

\[ Q'(s, a) = Q(s, a) + constant \times \hat{H}(s, a) \]

Outperforming:

On the metrics:

<table>
<thead>
<tr>
<th>TAMER-only</th>
<th>RL-only</th>
</tr>
</thead>
<tbody>
<tr>
<td>cumulative reward</td>
<td>only on ( \hat{H}_1 )</td>
</tr>
<tr>
<td>final performance</td>
<td></td>
</tr>
</tbody>
</table>
Complete successes

\[ a = \arg \max_a [Q(s, a) + \text{weight} \times \hat{H}(s, a)]. \]

and

\[ P(a = \arg \max_a [\hat{H}(s, a)]) = p. \] Otherwise original RL agent’s action selection mechanism is used.

Outperforming:

On the metrics:

<table>
<thead>
<tr>
<th>TAMER-only</th>
<th>RL-only</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="cumulative reward" /></td>
<td><img src="image2" alt="cumulative reward" /></td>
</tr>
<tr>
<td><img src="image3" alt="final performance" /></td>
<td><img src="image4" alt="final performance" /></td>
</tr>
</tbody>
</table>
Lessons

1. Optimistic vs. pessimistic initialization

Optimistic

Pessimistic

Bias initial actions toward $a_1$
Lessons

2. Biasing action selection was most effective

• better than shaping rewards
Summary

• background: TAMER Framework for learning from interactive shaping
• explored ways of combining TAMER with RL
• identified several successful techniques and some general lessons

• future: more domains, more RL algorithms