We have access to arbitrary amounts of data, and 54 stickers and 6 colors gives 324 inputs.

Algorithm

We modify AdaBoost.M2 to sample new data every timestep. Weak learners must output a confidence vector over classes: \( f : \{X, Y\} \rightarrow [0, 1] \).

AdaBoost.M2 turns \((x_i, y_i)\) into mislabelings \(\{(x_i, y_i)\}\) for every wrong \(y\). AdaBoost then expects weak learner \(f\) to discriminate \(y\), from \(y^\prime\):
\[
ploss(f, (x_i, y), y^\prime) = (1/2)(1 - f(x, y) - f(x, y^\prime))
\]

For any distribution \(D\) over mislabelings, require an edge over random guessing: \(\mathbb{E}[ploss(f, \cdot, y)] < \frac{1}{2}\)

**Algorithm 1 Refreshed AdaBoost**

1. **Input:** Dataset size \(n\), oracle \(Sample()\), \(p \in [0, 1]\)
2. **Init** \(\{\{x_i, y_i\}\} with \(n\) calls to \(Sample()\)
3. **for** \(i = 1\) to \(T\) **do**
4. **Train** weak learner \(f_i\) with distribution \(D_i\)
5. \(e_i \leftarrow \frac{1}{2\sum_{j}(y_i)D_i(x_i, y_i)ploss(f_i, (x_i, y), y_i)}\)
6. \(\alpha_i \leftarrow \log\frac{1}{e_i}\)
7. \(F_i \leftarrow \frac{1}{2\sum_{i=1}^{T}\alpha_if_i}\)

**Convergence Theorem**

\[
\mathbb{E}[0/1] \text{ loss of } F_T \leq (|Y| - 1) \prod_{t=1}^{T} 2\sqrt{1 - e_t}
\]

Prove by reduction to two class AdaBoost [2]. Still holds in expectation because the replaced data is sampled from the same distribution.

**Neural Net Training And Boosting**

The weak learner is a neural net. We train a single neural net over all \(t\), multiplying gradients by sample weight. Save snapshot of net for each \(t\). For efficiency, we only keep the models with largest \(\alpha_t\).

**Architecture Choice**

Initial experiments showed LSTMs outperforming RNNs and fully connected nets, even when all had the same number of parameters. Classification accuracy directly led to improved solving ability.

**Results**

Net trained with AdaBoost guidance did worse than baseline, and took longer to train.

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy</th>
<th>Run Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>72.26%</td>
<td>237 min</td>
</tr>
<tr>
<td>Boosted, (p = 0.8)</td>
<td>67.56%</td>
<td>366 min</td>
</tr>
<tr>
<td>Boosted, (p = 0)</td>
<td>67.91%</td>
<td>439 min</td>
</tr>
</tbody>
</table>

Baseline also has better solve percentages across the board. Some generalization, but solve rate drops quickly.

**Analysis**

Intuitively, AdaBoost should work better when:
- Weak learners are cheap to train and evaluate.
- Weak learners propose different outputs when given the same input.

Neural nets are not cheap to train, and working with the same neural net means the net is regularized against changing too much each iteration. Manual inspection showed neural nets tended to agree on best label.

Theoretically, approach will work with enough iterations, but it is too computationally expensive and there are not enough gains.

Current approach does not deal with fuzziness in episode labels well. Further work needs to address that multiple moves are valid.

**Extensions**

- Use reinforcement learning to optimize solve percentage. (RL better suited to discrete reward.)
- Curriculum learning - slowly increase \(K\) as net achieves performance.
- Adjust loss and labels if neural net predicts a valid solution that differs from ours.

**References**


**More Information**

- Code: github.com/alexirpan/rubik_research
- Email: alexirpan@berkeley.edu