Transforming task representations to allow deep learning models to perform novel tasks

Andrew K. Lampinen & James L. McClelland
Summary

Goal: enable zero-shot generalisation to novel tasks

Learn vector-based task representations

Learn “meta-mappings” (higher order tasks) that transform task representations

Demonstrate on regression, image classification, and reinforcement learning
Zero-shot task adaptation

If I tell you to lose at poker, you would do well at this task despite only trying to win in the past.

ML models can’t deal with this, especially if the task is the opposite.

Idea: condition behaviour on tasks, and learn relationships between tasks.

Can use natural language, or infer tasks using meta-learning.
Meta-Mappings

Learn to perform tasks

Learn mapping from task to task

Apply mapping for zero-shot adaptation

Tasks are input-output mappings, and so are meta-mappings, so can use same networks (same parameters) for both!
Input-Output Mappings

Tasks are I-O functions: image -> label, chessboard -> move

Meta-mappings are I-O functions: task -> task

Example: “lose” meta-mapping: win poker -> lose poker
Homoiconic programming languages are where programs can be manipulated in the same way as data.

Same hypernetwork takes task/meta-mapping embedding as input to create task network weights.

Future work could allow further recursion.
Constructing Task Representations

Language-based: use an RNN

Example-based: use a network that is set-invariant (e.g. uses max operator)
Performing Tasks

Domain-agnostic task network has weights from hypernetwork conditioned on task embedding.

Task network receives inputs from domain-specific perception network and outputs to domain-specific action network.

Trained end-to-end on task loss (regression, classification, etc.)
Constructing Meta-Mappings

Analogous to constructing basic task representations

Use the same networks/parameters, so everything maps to the same space

Trained by minimising L2 distance of task embedding with mapping versus target task embedding e.g.

\[ \text{min}_L^2(\text{lose(hearts)} - \text{lose_hearts}) \]
Transforming Tasks

Use the task network directly to perform meta-mappings

Used in meta-learning outer loop

(f) Transforming a task via a meta-mapping.
Training

Inference (black lines) and gradients (red lines)

Meta-mapping gradients were stopped at task embeddings because of computational bottleneck
Experiments

4 different domains/3 task types

Test zero-shot task generalisation e.g. lose(poker)

Test meta-mapping generalisation e.g. train R->B, G->Y, test R->Y

Test language-based generalisation

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Hold-out MMs</th>
<th>Lang. Comp.</th>
<th>Type</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Polynomials</td>
<td>✓</td>
<td></td>
<td>Regression</td>
<td>Vector (R^4)</td>
<td>Scalar (R)</td>
</tr>
<tr>
<td>Cards</td>
<td>✓</td>
<td>✓</td>
<td>Regression</td>
<td>Several-hot Bet values (R^3)</td>
<td></td>
</tr>
<tr>
<td>Visual concepts</td>
<td>✓</td>
<td>✓</td>
<td>Classification</td>
<td>50 x 50 image</td>
<td>Label ({0, 1})</td>
</tr>
<tr>
<td>RL</td>
<td>✓</td>
<td>✓</td>
<td>RL</td>
<td>91 x 91 image</td>
<td>Action Q-values (R^4)</td>
</tr>
</tbody>
</table>

Table 1. The contributions of our four sets of experiments. Our results span various computational paradigms and data types.
Polynomials

Task is polynomial regression, meta-mapping is addition / multiplication / squaring / permutation

New meta-mappings e.g. train on some permutations, evaluate on held-out permutations using example network
Card Games

Regress bet given cards (map state to action and reward)

36 training tasks (win/lose card games), test on losing poker

Example-based generalises, humans generalise, language-based does not (likely not strong enough inductive bias + little training data)
Visual Concepts

Binary classification of concept based on shape, colour and/or size e.g. triangle AND red

Meta-mappings for switching shape / colour

Example-based and language-based generalise

Meta-mapping generalisation improves with training samples
Reinforcement Learning

Pick up or push off target object (+ reward), distractor object (- reward)

18 training tasks, 2 test

Example-based generalises, language-based does not
Transfer Learning

Keep domain-specific (perception and action) networks fixed, optimise task embedding

Prevents interfering with prior knowledge!

Meta-mapping initialisation is best
Connections

Zero-shot language-based adaptation, meta-learning, task embeddings

Systematic, structured generalisation through learning

“our shared workspace for data points, tasks, and meta-mappings connects to ideas like the Global Workspace Theory of consciousness”

“modularity may not be built in [but] may result from the relationship among representations”