Learning to Discover Efficient Mathematical Identities

by Wojciech Zaremba, Karol Kurach, and Rob Fergus

ref: http://arxiv.org/abs/1406.1584
A toy example

Let’s consider two matrices $A$, $B$

$$\sum_{i,k} (AB)_{i,k} = \sum_{i} \sum_{j} \sum_{k} a_{i,j} b_{j,k}$$
A toy example

Let’s consider two matrices $A$, $B$

$$
\sum_{i,k} (AB)_{i,k} = \sum_i \sum_j \sum_k a_{i,j} b_{j,k}
$$

Naive computation takes $O(n^3)$. 
A toy example

Let’s consider two matrices $A$, $B$

$$\sum_{i,k} (AB)_{i,k} = \sum_i \sum_j \sum_k a_{i,j} b_{j,k}$$

Naive computation takes $O(n^3)$. Our framework found $O(n^2)$ computation
Overview

- How to represent computation
- How to search over computations
- Distributed representation of computation
Computation encoding $A\times B$

Symbolic representation $A\times B$ based on monomials
Computation encoding

\[ \text{sum}(A \ast B). \text{ Takes } O(n^3) \text{ time.} \]

\[ \sum \left( \begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{bmatrix} \begin{bmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{bmatrix} \right) = \left( \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} , \begin{bmatrix} a_{1,1} b_{1,1} \\ a_{1,2} b_{2,1} \\ a_{1,1} b_{1,2} \\ a_{1,2} b_{2,2} \\ a_{2,1} b_{1,1} \\ a_{2,2} b_{2,1} \\ a_{2,1} b_{1,2} \\ a_{2,2} b_{2,2} \end{bmatrix} \right) \]

Symbolic representation \( \text{sum}(A \ast B) \) based on monomials
Computation encoding

\[ \text{sum}(\text{sum}(A, 1)^*B) \]. Takes \( O(n^2) \) time

\[
\sum \left( \begin{bmatrix} a_{1,1} + a_{2,1} & a_{1,2} + a_{2,2} \\ b_{2,1} & b_{2,2} \end{bmatrix} \right) = \left\langle \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} a_{1,1}b_{1,1} \\ a_{1,2}b_{2,1} \\ a_{1,1}b_{1,2} \\ a_{1,2}b_{2,2} \\ a_{2,1}b_{1,1} \\ a_{2,2}b_{2,1} \\ a_{2,1}b_{1,2} \\ a_{2,2}b_{2,2} \end{bmatrix} \right\rangle
\]

Symbolic representation \( \text{sum}(\text{sum}(A, 1)^*B) \) based on monomials
Allowed computations

Grammar rules:

- matrix multiplication
- elementwise multiplication
- transposition
- sum, over columns and rows
- addition, multiplication by constant
- we can consider arbitrary bigger grammar ....

e.g. : (((sum((sum((A * (A')), 1)), 2)) * ((A * (((sum((A'), 1)) * A')'))')) * A)
Many computations are in this family

- E.g. finite Taylor expansion of any function ....
Many computations are in this family

- E.g. finite Taylor expansion of any function .... for instance, partition function of Restricted Boltzmann Machine (RBM)

\[
\sum_{v,h} \exp(v^T W h) = \sum_{k} \sum_{v,h} \frac{1}{k!} (v^T W h)^k
\]

\[v \in \{0, 1\}^n\]
\[h \in \{0, 1\}^m\]
Exact solution for $k=1$
(first term in Taylor series)

$$\sum_{v,h} v^T W h = 2^{n+m-2} \sum_{i,j} W_{i,j}$$

$v \in \{0, 1\}^n$

$h \in \{0, 1\}^m$

this is a polynomial computation vs exponential computation in the naive algorithm
Exact solution for \( k=2 \) (second term in Taylor series)

\[
\sum_{v,h}(v^TWh)^2 = 2^{n+m-4} \\
\sum_{i,j} W_{i,j}^2 + (\sum_{i,j} W_{i,j})^2 + \\
\sum_i (\sum_j W_{i,j})^2 + \sum_j (\sum_i W_{i,j})^2
\]

\( v \in \{0,1\}^n \)

\( h \in \{0,1\}^m \)

this is a polynomial computation vs exponential computation in the naive algorithm
How to find equivalent computations?
How to find equivalent computations?

Manual methods fail
(I have spent half a year on it).
Exact solution for $k=6$ (sneak preview) derived by our framework.
Maybe machines should be searching for patterns in computation
Overview

- How to represent computation
- How to search over computations
- Distributed representation of computation
Explosion of computation space

Polynomials of degree one in a matrix A:

\[ A, A^T, \sum_i A_{i,:}, \sum_j A_{:j}, \sum_{i,j} A, \sum_i A^T_{i,:}, \sum_j A^T_{:j} \]

Polynomials of degree two:

\[ A^2, (A^2)^T, A A^T, A^T A, \sum_i (A A^T)_{i,:}, \sum_{i,j} (A A^T)_{i,j}, \sum_i A^2_{i,:}, \sum_j A^2_{:j}, (\sum_{i,j} A)^2, \ldots \]

Space grows super-exponentially fast.
Prior over computation trees

- Explore space of computation efficiently
- Find equivalent expressions to the target one
  - But using operations with lower complexity

- Want to learn prior over sensible computations
  - Humans learn prior over proofs in mathematics
Searching over computation trees

**Scheduler** picks potential new expressions to append to current expressions

**Scorer** ranks each possibility (i.e. how likely they are to lead to the solution), using **prior**.

We want to **learn** a good scorer.
Scoring strategies

- naive scorer don’t use any prior. All computations are equally probable
- n-gram models
- learnt scorer (little bit about it at the end)
n-gram prior over trees

Exemplary intermediate solution:

Bi-grams:

Build n-grams distribution from solutions of simpler expressions

- Patterns that worked before might be useful
Experiments:
5 families of related problems

- $(\sum AA^T)_k$
- $(\sum (A \ast A)A^T)_k$
- Symmetric polynomials, e.g. $\sum_{i<j<k} A_i A_j A_k$
- RBM-1 $\sum_{v \in \{0,1\}^n} (v^T A)^k$
- RBM-2 $\sum_{v \in \{0,1\}^n, h \in \{0,1\}^n} (v^T A h)^k$
Family \( \text{sum}(A A^T)_k \)

Targets → Exemplary solution:

- \( \text{sum}(A^*A') \rightarrow (\text{sum}(((\text{sum}(A, 1)) \cdot (\text{sum}(A, 1))), 2)) \)
- \( \text{sum}(A^*A^*A) \rightarrow (\text{sum}((\text{sum}(A \cdot \text{sum}(((\text{sum}(A, 2)) \cdot A, 1))), 2), 1)) \)
- \( \text{sum}(A^*A^*A^*A') \rightarrow (\text{sum}((\text{sum}(A \cdot \text{sum}(((\text{sum}(A, 1)) \cdot (\text{sum}(A, 1))), 2)) \cdot A, 1))), 2)) \)
- \( \text{sum}(A^*A^*A^*A^*A) \rightarrow (\text{sum}((\text{sum}(A \cdot \text{sum}(((\text{sum}(A, 2)) \cdot A, 1))), 2)) \cdot A, 1))), 2)), 1)) \)
- ...
Family (RBM-1)_k

Targets → Exemplary solution:

- \[ \sum_{v \in \{0,1\}^n} \langle v, A \rangle \quad \rightarrow 16.0 \times (\text{sum}(\text{sum}(A, 2)), 1) \]

- \[ \sum_{v \in \{0,1\}^n} \langle v, A \rangle^2 \quad \rightarrow 8.0 \times (\text{sum}(((\text{sum}(A, 1)) \times (\text{sum}(A, 1))), 2)) + 8.0 \times ((\text{sum}(\text{sum}(A, 1)), 2)) \times (\text{sum}(\text{sum}(A, 1)), 2)) \]

- \[ \sum_{v \in \{0,1\}^n} \langle v, A \rangle^3 \quad \rightarrow 12.0 \times (\text{sum}((A \times (\text{sum}(((\text{sum}(\text{sum}(A, 2)), 1)) \times A, 1))), 2)), 1)) + 4.0 \times (\text{sum}(((\text{sum}(\text{sum}(A, 2)), 1)) \times ((\text{sum}(\text{sum}(A, 2)), 1)) \times (\text{sum}(A, 2)))))), 1)) \]

<table>
<thead>
<tr>
<th>Hardest possible example to solve</th>
<th>naive</th>
<th>1-gram</th>
<th>2-gram</th>
<th>3-gram</th>
<th>4-gram</th>
<th>5-gram</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardest possible example to solve</td>
<td>9</td>
<td>10</td>
<td>14</td>
<td>13</td>
<td>14</td>
<td>&gt;15</td>
</tr>
</tbody>
</table>
Overview

- How to represent computation
- How to search over computations
- Distributed representation of computation
The meaning of a word computation is described by the words computations accompanying it.
How we can represent a computation?

- Vector representation for every computation
  - e.g. $A^T = \text{vector}_1$, $\sum(A^T, 1) = \text{vector}_2$,

- Want to learn how to compose their vector representations
  - i.e. $((A^T)^T)^T \sim \text{vector}_1$, $\sum(A, 2)^T \sim \text{vector}_2$
Learnt representation with neural net

Recursive Neural network → RNN

(a) $(A \ast A)' \ast \text{sum}(A, 2)$

No understanding of underlying mathematical operators (no grounding)
Learnt representation with RNN
Recursive Neural network → RNN

(a) \((A \cdot A)' \ast \text{sum}(A, 2)\)

(b) \((A' \cdot A') \ast \text{sum}(A, 2)\)

No understanding of underlying mathematical operators (no grounding)
Task - classify expressions

Example from A class:

$$(((\text{sum}(\text{sum}(A \times (A')), 1)), 2)) \times (((A \times ((\text{sum}(A'), 1)) \times A'))) \times A)$$

Example from B class:

$$(((\text{sum}(A'), 1)) \times (A \times ((A') \times ((\text{sum}(A, 2)) \times ((\text{sum}(A'), 1)) \times A))))$$

From which class is this example?

$$(((\text{sum}(\text{sum}(A \times (A')), 1)), 2)) \times ((\text{sum}(A'), 1)) \times (A \times ((A') \times A))))$$
Performance - expression classification

Test accuracy

<table>
<thead>
<tr>
<th></th>
<th>Degree k = 3</th>
<th>Degree k = 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test accuracy</td>
<td>100%</td>
<td>95.3%</td>
</tr>
<tr>
<td>Number of classes</td>
<td>12</td>
<td>1687</td>
</tr>
</tbody>
</table>
Learnt representation - tricks

- Initialization as identity + noise (critical)
- ReLU (previously people used tanh)
- Curriculum learning
- Prediction matrix has x100 learning rate
- We update initial random vector of symbol
RNNs for a better discovery learning

- We have a real vector representation for any computations

- We use a linear classifier on such representation to train scorer
Family sum(AA^T)_k with RNN

RNN gives more diversified solutions (doesn’t just copy them), but it doesn’t perform as good as n-gram.

Targets → Exemplary solution of RNN:

- sum(A*A') → (sum((A * ((sum(A, 1))'))), 1))
- sum(A*A*A) → ((sum(A, 1)) * ((A') * (sum(A, 2))))
- sum(A*A*A*A') → (((sum(A, 1)) * (A') * A) * ((sum(A, 1))))

<table>
<thead>
<tr>
<th></th>
<th>naive</th>
<th>5-gram</th>
<th>RNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardest possible example to solve</td>
<td>10</td>
<td>&gt;15</td>
<td>~15</td>
</tr>
</tbody>
</table>
Summary

● Simple statistical priors over computations like n-gram allows the discovery of many new math formulae

● Use neural nets to map computational expressions to continuous vectors
  ○ Also use for formulae discovery
Future work

● Computations = Knowledge representation = Mathematical proofs = Programs = etc.
  ○ predictions on programs / program induction
  ○ explore space of mathematical proofs

● Replace recursive neural network with recurrent ?
Q&A

- How to represent computation
  - symbolic representation
- How to search over computations
- Distributed representation of computation
  - recursive networks
  - training tricks

Thank you. I am happy to take any question.