Efficient Algorithms

University of Zurich UZH

Prof. Dan Olteanu,
Dr. Ahmet Kara, Dr. Nils Vortmeier,
Haozhe Zhang

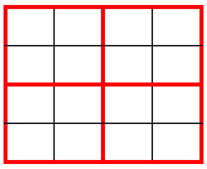
DaST
Data • (Systems+Theory)

Spring 2021 — Exercise Sheet 1 — 12.04.2021

- The solutions will be discussed on Monday 26.04.2021, 16:15-17:45 on Zoom
- Videos with solutions will be posted in OLAT after the exercise session

Exercise 1.1 [Sudoku]

(a) Consider the following 4×4 Sudoku board:



A solution to the above Sudoku assigns to each cell a digit from 1 to 4 such that

- no digit appears twice in any row,
- no digit appears twice in any column, and
- no digit appears twice in any 2×2 red bordered box.

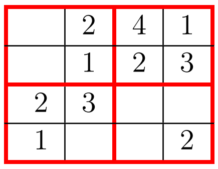
(i) Write an FAQ Φ_1 that computes all solutions to the above Sudoku. Give for each input factor of Φ_1 its set of tuples. Draw the hypergraph $\mathcal{H}_1$ of Φ_1.

(ii) Give $\rho^*(\mathcal{H}_1)$. Justify your answer.

(iii) Is $\mathcal{H}_1$ α-acyclic, β-acyclic, or free-connex α-acyclic?

(iv) How would you change Φ_1 to obtain an FAQ that decides whether the above Sudoku has at least one solution? Is the time to compute the changed Φ_1 lower than the time to compute Φ_1?

(b) Now, consider the following Sudoku board where some digits are preassigned:



	2	4	1
	1	2	3
2	3		
1			2

(i) Simplify the FAQ Φ_1 from Part (a) to an FAQ Φ_2 that computes all solutions to the above Sudoku with preassigned digits. Give for each input factor of Φ_2 its set of tuples. Draw the hypergraph $\mathcal{H}_2$ of Φ_2.

(ii) Give $\rho^*(\mathcal{H}_2)$. Justify your answer.

(iii) Is $\mathcal{H}_2$ α-acyclic, β-acyclic, or free-connex α-acyclic?

(c) Now, consider the following Sudoku board:

3	2		
4	1		
		1	4
		3	2

(i) Simplify the FAQ Φ_1 from Part (a) to an FAQ Φ_3 that computes all solutions to the above Sudoku. Draw the hypergraph $\mathcal{H}_3$ of Φ_3.

(ii) Give $\rho^*(\mathcal{H}_3)$. Justify your answer.

(iii) Is $\mathcal{H}_3$ α-acyclic, β-acyclic, or free-connex α-acyclic?

Exercise 1.2 [SQL]
In this exercise we design FAQs that are equivalent to SQL queries. Assume we have three tables R, S, and T in our relational database. Table R has attributes A and B, table S has attributes B and C, and table T has attributes C and D. The domains of all attributes are natural numbers.

(a) Consider the SQL query:

```
SELECT C, MAX(B)
FROM R NATURAL JOIN S NATURAL JOIN T
GROUP BY C
```

Give an FAQ Φ that is equivalent to the above SQL query, i.e., for any R, S, and T, Φ maps each C-value c to the maximum B-value paired with c in the natural join of R, S, and T.

(b) Consider now the SQL query:

```
SELECT C, MAX(B), SUM(B)
FROM R NATURAL JOIN S NATURAL JOIN T
GROUP BY C
```

Give a *single* FAQ Φ that is equivalent to the above SQL query.

Exercise 1.3 [Quantified Formulas]
Consider the following first-order formulas for some relations P, Q, and R. Write for each formula an FAQ that counts the number of satisfying assignments to the free variables of the formula.

(a) $\varphi(x_1, x_2, x_3, x_4, x_5) = P(x_1, x_3) \wedge Q(x_3, x_4, x_5) \wedge R(x_1, x_2, x_5)$

(b) $\varphi(x_1, x_2, x_4) = \exists x_3 \exists x_5 P(x_1, x_3) \wedge Q(x_3, x_4, x_5) \wedge R(x_1, x_2, x_5)$

(c) $\varphi(x_1, x_2) = \exists x_3 \forall x_4 \exists x_5 P(x_1, x_3) \wedge Q(x_3, x_4, x_5) \wedge R(x_1, x_2, x_5)$

Exercise 1.4 [Einsum]
Einsum notation is a convenient way of expressing operations on tensors, i.e. matrices of higher degrees. Popularised by Albert Einstein, the notation is nowadays supported by machine learning libraries like numpy, TensorFlow, and PyTorch. We present an example following the conventions in numpy (for a basic guide to einsum, see [1]). Let us say we want to multiply two matrices $A \in \mathbb{R}^{m\times n}$

and $B \in \mathbb{R}^{n \times q}$ resulting in a matrix $C \in \mathbb{R}^{m \times q}$. Each cell C_{ij} with $i \in [m]$ and $j \in [q]$ can be expressed as

$$C_{ij} = \sum_k A_{ik} \cdot B_{kj}.$$

In einsum notation, we can express C as

$$C = \texttt{einsum}('ik, kj \to ij', A, B)$$

The letters before the first comma (here, i and k) are the indices of A (i is the index for the rows and k for the columns) and the letters following the first comma (here, k and j) are the indices of B. A letter that appears before and after the first comma (here, k) signalises that only those cells of A and B that agree on the index corresponding to that letter will be multiplied together. The letters following the arrow (here, i and j) are the indices of the output matrix C. If an index is omitted from the output (here, k) this means that values along that axis will be summed up. Assume now that we want to sum over each row in C. This yields a vector $v \in \mathbb{R}^m$ with

$$v_i = \sum_j \sum_k A_{ik} \cdot B_{kj} \quad \text{for } i \in [m].$$

Using einsum, the vector v can be defined by

$$v = \texttt{einsum}('ik, kj \to i', A, B).$$

In this exercise we will formulate einsum expressions as FAQs. Recall that a tensor A of degree d can be represented by a factor ψ_A with d variables such that $\psi_A(i_1, \ldots, i_d) = A_{i_1 \ldots i_d}$. In particular, a matrix (tensor of degree 2) can be represented by a factor with two variables.

(a) Formulate each of the following einsum expressions as an FAQ.

(i) Given matrices $A \in \mathbb{R}^{m \times n}$, $B \in \mathbb{R}^{n \times p}$, and $C \in \mathbb{R}^{p \times m}$, the einsum expression

$$t = \texttt{einsum}('ij, jk, ki \to ', A, B, C)$$

computes the trace of the product of A, B, and C.

(ii) TreeQN is an end-to-end differentiable tree-structured neural network for deep reinforcement learning [4]. Given a state representation $z_\ell \in \mathbb{R}^k$ at a layer ℓ of such a network and a transition function $W^a \in \mathbb{R}^{k \times k}$ per action $a \in \mathcal{A}$, the next state representation can be calculated by

$$z^a_{l+1} = z_l + \texttt{tanh}(W^a z_l).$$

If we want to do this efficiently for a batch $Z \in \mathbb{R}^{b \times k}$ of b k-dimensional state representations and for all actions in $\mathcal{A}$ at the same time, we can arrange these functions in a tensor $W \in \mathbb{R}^{|\mathcal{A}| \times k \times k}$ and use einsum to calculate the next-state representations (for detailed pseudocode, see [2]):

$$A = \texttt{einsum}('bk, aki \to bai', Z, W)$$

(iii) The literature proposes neural models to recognize entailment between two sentences in natural language [5]. A word-by-word attention mechanism to check the semantic relationship between two sentences in such a model computes the following matrices and vectors:

$$\begin{aligned} M_t &= \texttt{tanh}(W^y Y + (W^h h_t + W^r r_{t-1}) \otimes e_L) \\ \alpha_t &= \texttt{softmax}(w^T M_t) \\ r_t &= Y \alpha_t^T + \texttt{tanh}(W^t r_{t-1}) \end{aligned}$$

where $Y \in \mathbb{R}^{k\times\ell}$, $W^y, W^h, W^r, W^t \in \mathbb{R}^{k\times k}$, $h_t, r_{t-1}, w \in \mathbb{R}^k$, $e_L \in \mathbb{R}^\ell$ and $\otimes$ stands for outer product.

A batch implementation requires the computation of the following einsum expressions (for detailed pseudocode, see [2]):

$$\begin{aligned}
A &= \texttt{einsum}('ik, kl \to il', h_t, W^h) + \texttt{einsum}('ik, kl \to il', r_{t-1}, W^r)\\
B &= \texttt{einsum}('ijk, kl \to ijl', Y, W^y)\\
C &= \texttt{einsum}('ijk, k \to ij', M_t, w)\\
D &= \texttt{einsum}('ijk, ij \to ik', Y, C)\\
E &= \texttt{einsum}('ij, jk \to ik', r_{t-1}, W^t)
\end{aligned}$$

where $h_t, r_{t-1} \in \mathbb{R}^{n\times k}$, $Y, M_t \in \mathbb{R}^{n\times m\times k}$, and W^h, W^y, W^t, and w are defined as above.

(b) There are accounts of the numpy's implementation of einsum being rather poor [3]. It was noticed that the performance of

$$A = \texttt{einsum}('ik, km, im \to i', X, C, X)$$

is poorer than when breaking it down into two steps:

$$\begin{aligned}
B &= \texttt{einsum}('ik, km \to im', X, C)\\
A &= \texttt{einsum}('ik, ik \to i', B, X)
\end{aligned}$$

Formulate both strategies in FAQ. Using the leapfrog triejoin algorithm introduced in the lecture, which of the above strategies would perform better?

Exercise 1.5 [Width Measures]

(a) Give a class **H** of α-acyclic hypergraphs such that for each $m \in \mathbb{N}_{\geq 1}$, **H** contains a hypergraph $\mathcal{H}$ with m hyperedges and $\rho(\mathcal{H}) - \texttt{htw}(\mathcal{H}) = \rho^*(\mathcal{H}) - \texttt{fhtw}(\mathcal{H}) = m - 1$.

(b) Give a class **H** of cyclic hypergraphs such that for each $m \in \mathbb{N}_{\geq 3}$, **H** contains a hypergraph $\mathcal{H}$ with m hyperedges and $\rho(\mathcal{H}) - \texttt{htw}(\mathcal{H}) \geq \lfloor \frac{m}{2} \rfloor - 2$.

(c) Give the fractional hypertree width of

  (i) a Loomis-Whitney query of degree n,

  (ii) a Clique query of degree n.

(d) For each of the following pairs of widths w_1 and w_2, can you find two hypergraphs $\mathcal{H}_1$ and $\mathcal{H}_2$ such that: $w_1(\mathcal{H}_1) > w_2(\mathcal{H}_1)$ and $w_1(\mathcal{H}_2) < w_2(\mathcal{H}_2)$? Justify your answers.

  (i) ρ and ρ^*

  (ii) ρ and `fhtw`

  (iii) ρ^* and `htw`

  (iv) `htw` and `fhtw`

References

[1] Basic guide to einsum: https://ajcr.net/Basic-guide-to-einsum/.

[2] Einsum is all you need - Einstein summation in deep learning. Blog entry by Tim Rocktäschel: https://rockt.github.io/2018/04/30/einsum.

[3] Stack Overflow post - Why is numpy's einsum slower than numpy's built-in functions?: https://stackoverflow.com/questions/20149201/why-is-numpys-einsum-slower-than-numpys-built-in-functions.

[4] Gregory Farquhar, Tim Rocktäschel, Maximilian Igl, and Shimon Whiteson. TreeQN and ATreeC: Differentiable tree-structured models for deep reinforcement learning. In ICLR 2018.

[5] Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomás Kociský, and Phil Blunsom. Reasoning about entailment with neural attention. In ICLR 2016.