Labelled Modes

A New Notation for Tensor Networks

Simon Etter

University of Warwick

GAMM Annual Meeting
7th March 2017

Introduction

Mathematician's style goals

- Precise.
- Concise.
- General.

The tensor network notation dilemma

- Tensor trains: precise & concise, but not general.
- Hierarchical Tucker: general, but not precise or concise.

Introduction

Furthermore

- Even hierarchical Tucker not really general:
  Requires rooted binary tree, free modes only in leaves.
- Even tensor trains not really precise and concise:

$$x(i_1, \ldots, i_d) = \sum_{\alpha_1} \cdots \sum_{\alpha_{d-1}} x_1(i_1, \alpha_1)\, x_2(\alpha_1, i_2, \alpha_2) \ldots x_d(\alpha_{d-1}, i_d).$$

- Coding is hard:

```
x = x[1]
n = n[1]
for k = 2:d
    x = reshape(x, (n,r[k])) *
        reshape(x[k], (r[k],n[k]*r[k+1]))
    n *= n[k]
end
```

Introduction

Example

- Scientist counting fraction of species with particular features.
- Natural representation: $p(\#\text{extremities} = x, \#\text{eyes} = y, \ldots)$
- Tensor representation: $p(i_1, \ldots, i_d)$

Example

- Consider two tensors $A \in \mathbb{R}^{4\times 5\times 6}$, $B \in \mathbb{R}^{5\times 7\times 8}$.
- Only way to contract: 2nd mode of A with 1st mode of B.
- Order of modes in result follows from mode sizes.
- Hence, $C = AB$ contains same information as

$$C(a, c, d, e) = \sum_b A(a, b, c)\, B(b, d, e).$$

Labelled Modes Tensors

Generalised tuple

Let D be a finite set and $(A_k)_{k\in D}$, a family of sets.

- A *tuple* $t \in \bigtimes_{k\in D} A_k$ is a function

$$t : D \to \bigcup_{k\in D} A_k \quad \text{such that} \quad t_k \in A_k.$$

- Traditional tuples: $D = \{1, \ldots, n\}$.

Tensor

A *tensor* x is a function mapping integer tuples to scalars,

$$\mathbb{K}(D) := \left\{ x : \bigtimes_{k\in D} [n_k] \to \mathbb{K} \right\}, \quad \text{where} \quad [n_k] := \{1, \ldots, n_k\}.$$

Labelled Modes Tensors

This construction is natural and intuitive!
Recall example of scientist counting species.

- Data naturally represented as $p \in \mathbb{R}(\{\#\text{extremities}, \#\text{eyes}\})$.
- We can access elements using

$$p(i_{\#\text{extremities}} = 4, i_{\#\text{eyes}} = 2) = p(i_{\#\text{eyes}} = 2, i_{\#\text{extremities}} = 4)$$

Labelled Modes Tensors

Mode product

Let M, K, N be mutually disjoint mode sets.
Given two tensors $x \in \mathbb{K}(M \cup K)$, $y \in \mathbb{K}(K \cup N)$, their *mode product* $z := xy$ is the tensor $z \in \mathbb{K}(M \cup N)$ such that

$$z(i_M \times i_N) = \sum_{i_K} x(i_M \times i_K)\, y(i_K \times i_N).$$

Example

Recall example of tensor contraction. It now becomes

$$A \in \mathbb{K}(\{a, b, c\}), \quad B \in \mathbb{K}(\{b, d, e\}), \quad AB \in \mathbb{K}(\{a, c, d, e\})$$

$$(AB)(i_{\{a,c\}} \times i_{\{d,e\}}) = \sum_{i_b} A(i_{\{a,c\}} \times i_{\{b\}})\, B(i_{\{b\}} \times i_{\{d,e\}}).$$

Tree Tensors

Mode tree

A triplet (V, E, D) where (V, E) is a tree and D a function mapping vertices to disjoint mode sets.

$$\begin{tikzcd}
\{a\} \arrow[dr, dash] & & & \{e\} \\
\{b\} \arrow[r, dash] & \{\} \arrow[r, dash] & \{d\} \arrow[ur, dash] \arrow[dr, dash] & \\
\{c\} \arrow[ur, dash] & & & \{f, g\}
\end{tikzcd}$$

Tree tensor

A tuple of tensors

$$x \in \bigtimes_{v \in V} \mathbb{K}(E(v) \cup D(v)).$$

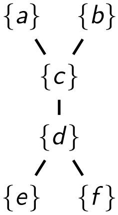
Contracting the network is trivial:

$$x = \prod_{v \in V} x_v.$$

TreeTensors.jl

Create mode tree and tree tensor:

```
julia> using Tensors, TreeTensors;
julia> mtree = ModeTree([Mode(:c,2)],
           ModeTree([Mode(:a,2)]),
           ModeTree([Mode(:b,2)]),
           ModeTree([Mode(:d,2)],
               ModeTree([Mode(:e,2)]),
               ModeTree([Mode(:f,2)]),
           )
       );
julia> x = rand(mtree,2);
```



Contract network:

```
julia> prod(values(x.tensors))   # Lacks order
Tensor{Float64}([#= Modes :a,:b,:d,:c,:f,:e =#])

julia> contract(x)   # Contract leaves-to-root
Tensor{Float64}([#= Modes :c,:a,:b,:f,:d,:e =#])
```

Conclusion

Labelled Modes (personal review)

- No notational blinkers.
- Shorter and more expressive formulae.
- Easy and bulletproof coding.

Try it yourself, it's free!

Conclusion

Material

- Slides: `homepages.warwick.ac.uk/student/S.Etter/`
- `Tensors` and `TreeTensors` packages: `github.com/ettersi`
- Publication:
  S. Etter, *Parallel ALS Algorithm for Solving Linear Systems in the Hierarchical Tucker Representation*, SIAM Journal on Scientific Computing

Acknowledgements
Thanks to Vladimir Kazeev, Robert Gantner, Christoph Schwab and Christoph Ortner for their help with this work!

Mode Tags

Mode tag
A symbol t such that $t(k)$ introduces a new mode based on k.

Row and Column mode tags
Tags R and C with special multiplication rules.

- $R(k)$ only multiplies k and $C(k)$ appearing on the left.
- $C(k)$ only multiplies k and $R(k)$ appearing on the right.
- If multiplication produces unpaired $R(k)/C(k)$, rename to k.

Write $D^2 := R(D) \cup C(D)$.

Example

- Operator $A \in \mathbb{K}(D^2)$, vector $x \in \mathbb{K}(D)$.
- Without last rule: $Ax \in \mathbb{K}(R(D))$.
- With last rule: $Ax \in \mathbb{K}(D)$.

Orthogonalisation

Maths paraphrased from [Ose11]

for $k = 1$ to $d-1$ **do**

$$[q(\beta_k i_k; \beta_{k+1}), r(\beta_{k+1}, \alpha_{k+1})] := \mathrm{QR}(x_k(\beta_k i_k; \alpha_{k+1}))$$
$$x_k = q$$
$$x_{k+1} := r \times_1 x_{k+1}$$

end for

Maths in labelled modes notation

for edge $v - p$ in leaves-to-root order **do**

$$(q, r) := \mathrm{QR}_{v-p}(x_v)$$
$$x_v := q$$
$$x_p := r\, x_p$$

end for

$$\boxed{\begin{array}{l} q \in \mathbb{K}(D(v) \cup E(v)) \\ r \in \mathbb{K}(\{v - p\}^2) \end{array}}$$

[Ose11] I. V. Oseledets. *Tensor Train Decomposition*. SIAM Journal on Scientific Computing (2011)

Orthogonalisation

Code

```
for (v,p) in edges(x, leaves_to_root)
    (q,r) = qr(x[v], PairSet(v,p))
    x[v] = q
    x[p] = r*x[p]
end
```

Truncation

Maths

for vertex v in root-to-leaves order **do**
    for child c of v **do**
        $(b, s_{v-c}, d) = \mathrm{SVD}_{v-c}(x_v)$
        $x_v = b\,\mathrm{diag}(s_{v-c})$
        $x_c = \mathrm{diag}(s_{v-c})\,d\,x_c$
        Truncate s_{v-c}
    end for
    Truncate x_v
    for child c of v **do**
        $x_v = \mathrm{diag}(s_{v-c}^{-1})\,x_v$
    end for
end for

Truncation

Code

```
s = Dict{PairSet{Tree}, Tensor{real(scalartype(x))}}()
for (v,p) in edges_with_root(x, root_to_leaves)
    for c in children(v,p)
        e = PairSet(v,c)
        b,s[e],d = svd(x[v], e, maxrank())
        x[c] = scale(s[e],d)*x[c]
        x[v] = scale(b,s[e])
        s[e] = resize(s[e], Dict(e => rank(s[e])))
    end
    x[v] = resize(x[v],
        Dict(e => length(s[e]) for e in neighbor_edges(v)))
    x[v] = scale(x[v],
        [1./s[e] for e in child_edges(v,p)]...)
end
```