Biblio :

Differentiable Programming Tensor Networks

↑ (hyperlink)

COMPUTING GRADIENTS OF TENSOR NETWORKS

(I) Jacobian and backpropagation

- $f : \mathbb{R}^n \to \mathbb{R}$ Gradient $\nabla_\theta f = \begin{pmatrix} \partial f/\partial \theta_1 \\ \vdots \\ \partial f/\partial \theta_n \end{pmatrix}$ for each $\theta \in \mathbb{R}^n$

- $f : \mathbb{R}^n \to \mathbb{R}^h$ Jacobian $\frac{\partial f}{\partial \theta} = \left(\frac{\partial f(\theta)_i}{\partial \theta_j} \right)_{i,j} \in \mathbb{R}^{h \times n}$ for each $\theta \in \mathbb{R}^n$

- Backpropagation / Automatic Differentiation (reverse mode).

Computational Graph $\theta \to T^1 \to T^2 \to T^3 \to \mathcal{L}$ (where $X \to Y$ means that Y is a function of X)

$\mathbb{R}^m \to \mathbb{R}^{d_1} \to \mathbb{R}^{d_2} \to \mathbb{R}^{d_3} \to \mathbb{R}$

We want to compute $\frac{\partial \mathcal{L}}{\partial \theta}$.

Chain rule: $\frac{\partial \mathcal{L}}{\partial \theta} = \frac{\partial \mathcal{L}}{\partial T_3} \frac{\partial T_3}{\partial T_2} \frac{\partial T_2}{\partial T_1} \frac{\partial T_1}{\partial \theta}$

$\bar{\theta}$: $1 \times m$; $1 \times d_3$; $d_3 \times d_2$; $d_2 \times d_1$; $d_1 \times m$

We define the adjoint: $\overline{T} = \frac{\partial \mathcal{L}}{\partial T}$

$\overline{T_3} = \frac{\partial \mathcal{L}}{\partial T_3}$

$\overline{T_2} = \frac{\partial \mathcal{L}}{\partial T_2} = \frac{\partial \mathcal{L}}{\partial T_3} \frac{\partial T_3}{\partial T_2} = \overline{T_3} \frac{\partial T_3}{\partial T_2}$

$\overline{T_1} = \frac{\partial \mathcal{L}}{\partial T_1} = \frac{\partial \mathcal{L}}{T_2} \frac{\partial T_2}{\partial T_1} = \overline{T_2} \frac{\partial T_2}{\partial T_1}$

$\bar{\theta} = \cdots = \overline{T_1} \frac{\partial T_1}{\partial \theta}$

$\hookrightarrow$ $\overline{T^j} = \overline{T^{j+1}} \frac{\partial T^{j+1}}{\partial T^j}$

More complex graphs: $\theta \to T_1 \nearrow T_2 \searrow T_4 \to \mathcal{L}$, $T_1 \searrow T_3 \nearrow T_4$

$$\overline{T_i} = \sum_{j \,:\, \text{child of } i} \overline{T_j} \frac{\partial T_j}{\partial T_i}$$

ex: $\overline{T_1} = \frac{\partial \mathcal{L}}{\partial T_1} = \frac{\partial \mathcal{L}}{\partial T_2}\frac{\partial T_2}{\partial T_1} + \frac{\partial \mathcal{L}}{\partial T_3}\frac{\partial T_3}{\partial T_1}$

$= \overline{T_2}\frac{\partial T_2}{\partial T_1} + \overline{T_3}\frac{\partial T_3}{\partial T_1}$

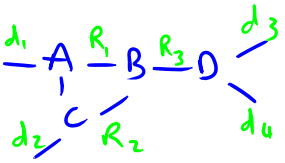
- $f : \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_N} \to \mathbb{R}^{m_1 \times m_2 \times \cdots \times m_P}$ $\left(\cong \mathbb{R}^{n_1 n_2 \cdots n_N} \to \mathbb{R}^{m_1 m_2 \cdots m_P}\right)$

Jacobian Tensor $\quad \frac{\partial f(T)}{\partial T} = \left(\frac{\partial f(T)_{i_1 \cdots i_P}}{\partial T_{j_1 \cdots j_N}}\right)_{i_1 \cdots i_P j_1 \cdots j_N} \in \mathbb{R}^{m_1 \times \cdots \times m_P \times n_1 \times \cdots \times n_N}$

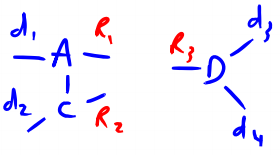
II Jacobian of Tensor Networks

Let W be a tensor given as a tensor network where G is a core tensor appearing only once. Then $\frac{\partial W}{\partial G}$ is simply obtained by deleting G in the tensor network.

Ex: • W =

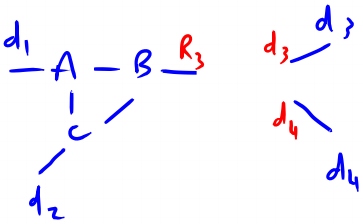


$\frac{\partial W}{\partial B}$ =



$d_1 \times d_2 \times d_3 \times d_4 \times R_1 \times R_2 \times R_3$

$\frac{\partial W}{\partial D}$ =



• Rederiving classical matrix / vector gradients:

$$\frac{\partial \langle u, v \rangle}{\partial u} = \partial(u - v)/\partial u = -v = v$$

$$\frac{\partial Ax}{\partial x} = \partial(-A-x)/\partial x = -A- \quad = A$$

$$\frac{\partial x^T A x}{\partial A} = \partial(x-A-x)/\partial A = x- \quad -x \quad = xx^T$$

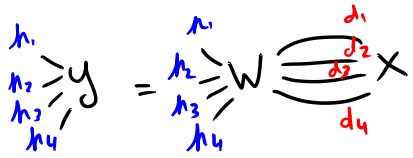
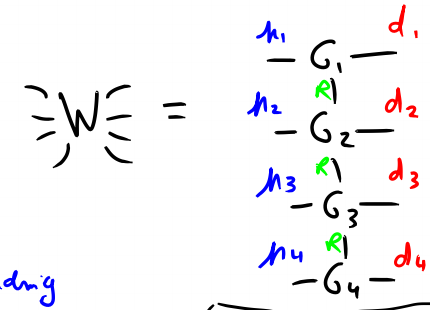
$$\frac{\partial \mathrm{Tr}(A)}{\partial A} = \partial(\,\textcircled{A}\,)/\partial A = \quad \cap \quad = I$$

$$\frac{\partial Ax}{\partial A} = \partial(-A-x)/\partial A = \quad - \quad -x \cong I \circ x$$

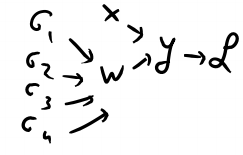
If the core tensor G appears k times in the tensor network of W, then $\frac{\partial W}{\partial G}$ is given the sum of k copies of W where a different occurence of G is deleted in each copy.

Ex: $\frac{\partial\, x^T A x}{\partial x} = \partial(x_1 A_2 x)/\partial x = x_1 A_2 + x_1 A_2 = (A + A^T)x$

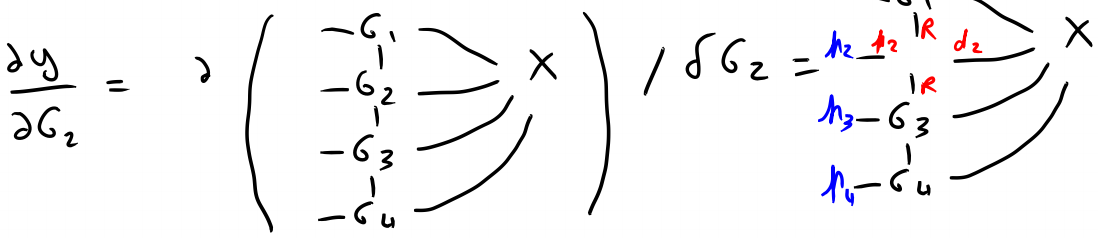
Applications

TENSORIZING NEURAL NETWORKS :  where 

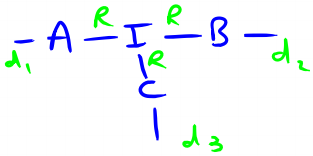
↳ We want to compute $\frac{\partial \ell}{\partial G^2} = \frac{\partial \ell}{\partial y}\frac{\partial y}{\partial G_2}$ ← some loss depending on W

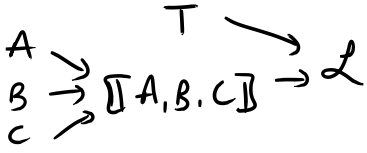
Comp. Graph 

We need to compute $\frac{\partial y}{\partial G_2} \in \mathbb{R}^{n_1 \times n_2 \times n_3 \times n_4 \times R \times n_2 \times d_2 \times R}$. We have :

$\frac{\partial y}{\partial G_2} =$ 

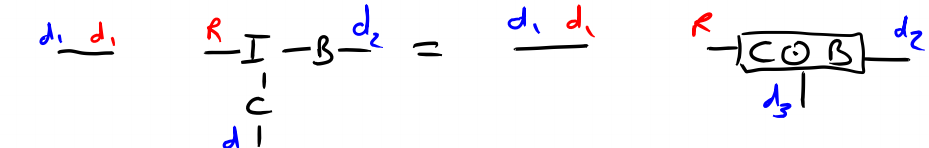
CP-DECOMPOSITION with GRADIENT DESCENT : $\ell = \| T - [\![A,B,C]\!] \|_F^2$ ($T \in \mathbb{R}^{d_1 \times d_2 \times d_3}$)



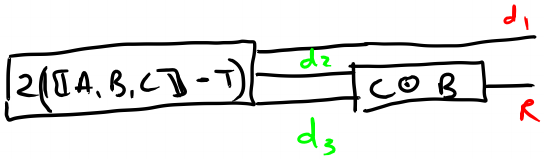
Comp. Graph : 

We want to compute $\frac{\partial \ell}{\partial A} = \frac{\partial \ell}{\partial [\![A,B,C]\!]}\frac{\partial [\![A,B,C]\!]}{\partial A}$

$\frac{\partial \ell}{\partial [\![A,B,C]\!]} = 2([\![A,B,C]\!] - T)$ $\quad 1 \times d_1 \times d_2 \times d_3$

$\frac{\partial [\![A,B,C]\!]}{\partial A}$ $\quad d_1 \times d_2 \times d_3 \times d_1 \times R$ $=$ 

$$\frac{\partial \mathcal{L}}{\partial A} = \frac{\partial \mathcal{L}}{\partial [\![A,B,C]\!]} \frac{\partial [\![A,B,C]\!]}{\partial A} =$$



$$= 2\left(T - [\![A,B,C]\!]\right)_{(1)} (C \odot B)$$

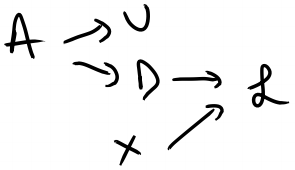
$d_1 \times d_2 d_3$ $\quad$ $d_2 d_3 \times R$

(III) Some results on adjoints of SVD / eigen / QR decompositions

Tensor network algorithm relies on operations like rounding:

close to the original with $\hat{R} < R$

We want to backprop through matrix decomposition, e.g.



where $A = UDU^T$ is an eig. decomposition.

↳ We need to compute e.g. $\overline{D} = \frac{\partial \mathcal{L}}{\partial D}$ $\quad$ $\overline{A} = \frac{\partial \mathcal{L}}{\partial A} = \overline{D} \frac{\partial D}{\partial A}$

(I) Eigen decomposition

$$\overline{A} = U\left[\overline{D} + F \odot (U^T\overline{U} - \overline{U}^T U)/2\right]U^T, \qquad (3)$$

↳ $F_{ij} = (d_j - d_i)^{-1}$ if $i \neq j$

(II) SVD

$$\overline{A} = \frac{1}{2}U\left[F_+ \odot \left(U^T\overline{U} - \overline{U}^T U\right) + F_- \odot \left(V^T\overline{V} - \overline{V}^T V\right)\right]V^T$$
$$+ U\overline{D}V^T + (I - UU^T)\overline{U}D^{-1}V^T + UD^{-1}\overline{V}^T(I - VV^T), \quad (4)$$

(III) QR

$$\overline{A} = \left[\overline{Q} + Q\,\mathtt{copyltu}(M)\right]R^{-T}, \qquad (5)$$

where $M = R\overline{R}^T - \overline{Q}^T Q$ and the `copyltu` function generates

IV Generalized CP decomposition

GENERALIZED CANONICAL POLYADIC TENSOR DECOMPOSITION*

DAVID HONG†, TAMARA G. KOLDA‡, AND JED A. DUERSCH§

↳ Likelihood maximization & generalized linear model.

Observations/Data : $X \in \mathbb{R}^{m_1 \times \cdots \times m_d}$

Parameterized PDF : $X_{i_1 \ldots i_d} \sim p(X_{i_1 \cdots i_d} \mid \theta_{i_1 \cdots i_d})$

where $\ell(\theta_{i_1 \cdots i_d}) = M_{i_1 \cdots i_d}$

$\ell : \mathbb{R} \to \mathbb{R}$ is an invertible link function connecting the model parameters $M_{i_1 \cdots i_d}$ to the "natural" parameter of the distribution.

Model parameters : $M \in \mathbb{R}^{m_1 \times \cdots \times m_d}$

Goal : Find M that maximizes the likelihood of the observed entries of X.

↳ $\Omega \subseteq [m_1] \times \cdots \times [m_d]$

$$\max_M \mathcal{L}(M; X, \Omega) \equiv \prod_{(i_1 \cdots i_d) \in \Omega} p(X_{i_1 \cdots i_d} \mid \theta_{i_1 \cdots i_d})$$

$$\text{where } \ell(\theta_{i_1 \cdots i_d}) = M_{i_1 \cdots i_d}$$

→ We only have (at most) one observation for each entry of X!

We will assume a low rank structure of M (interdependence between the $\theta_{i_1 \cdots i_d}$'s)

↳ low CP rank of M.

Reformulation of optimization problem :

$$\min_M F(M; X, \Omega) = \sum_{(i_1 \cdots i_d) \in \Omega} f(X_{i_1 \cdots i_d}, M_{i_1 \cdots i_d}) \quad \text{s.t. } \operatorname{rank}_{CP}(M) \leq R$$

$$\text{where } f(x, m) = -\log p(x \mid \ell^{-1}(m))$$

↑ "loss function" (assumed differentiable)

Examples of data distribution

① Gaussian distribution (Gaussian MLE $\Leftrightarrow$ Mean squared error)

$$X_{i,\dots,id} = M_{i,\dots,id} + \varepsilon_{i,\dots,id} \quad \text{where } \varepsilon_{i,\dots,id} \sim \mathcal{N}(0, \sigma^2)$$

(σ^2: constant)

Equivalently: $X_{i,\dots,id} \sim \mathcal{N}(\theta_{i,\dots,id}, \sigma^2)$ where $\theta_{i,\dots,id} = M_{i,\dots,id}$

the link function is the identity! $f(\theta) = \theta$

Simple derivation:

$$f(x, m) = -\log \mathcal{N}(x; m, \sigma^2) = \frac{(x-m)^2}{2\sigma^2} + \frac{1}{2}\log(2\pi\sigma^2)$$

↳ ignoring σ^2 and constants $\boxed{f(x, m) = (x-m)^2}$

$\Rightarrow$ ★ is then equivalent to CP decomposition.

② Bernoulli distribution.

If $X_{i,\dots,id} \in \{0,1\}$ (binary data), we can assume $X_{i,\dots,id} \sim \mathcal{B}(\theta_{i,\dots,id})$

$$h(x|\theta) = \theta^x (1-\theta)^{1-x} \quad , x \in \{0,1\}, \ \theta \in [0,1]$$

link function?

- identity $\rightsquigarrow$ we need to constrain $m \in [0,1]$
- odds ratio: $m = f(\theta) = \frac{\theta}{1-\theta}$

  $\rightsquigarrow$ only need to enforce $m \geq 0$
- log odds ratio:

  $m = f(\theta) = \log\left(\frac{\theta}{1-\theta}\right)$

  ↳ $\theta = \frac{e^m}{1+e^m}$, $1-\theta = \frac{1}{1+e^m}$

$$\dots \quad f(x, m) = -\log h(x|\theta) = \underbrace{-x\log\theta - (1-x)\log(1-\theta)}_{\text{cross-entropy}}$$

$$\boxed{f(x, m) = \log(1+e^m) - xm}$$

③ Other distributions

Table 1: Statistically-motivated loss functions. Parameters in blue are assumed to be constant. Numerical adjustments are indicated in red.

Distribution	Link function	Loss function	Constraints
$\mathcal{N}(\mu,\sigma)$	$m=\mu$	$(x-m)^2$	$x,m\in\mathbb{R}$
Gamma(k,σ)	$m=k\sigma$	$x/(m+\epsilon)+\log(m+\epsilon)$	$x>0, m\geq 0$
Rayleigh(θ)	$m=\sqrt{\pi/2}\,\theta$	$2\log(m+\epsilon)+(\pi/4)(x/(m+\epsilon))^2$	$x>0, m\geq 0$
Poisson(λ)	$m=\lambda$	$m-x\log(m+\epsilon)$	$x\in\mathbb{N}, m\geq 0$
	$m=\log\lambda$	e^m-xm	$x\in\mathbb{N}$, $m\in\mathbb{R}$
Bernoulli(ρ)	$m=\rho/(1-\rho)$	$\log(m+1)-x\log(m+\epsilon)$	$x\in\{0,1\}, m\geq 0$
	$m=\log(\rho/(1-\rho))$	$\log(1+e^m)-xm$	$x\in\{0,1\}$, $m\in\mathbb{R}$
NegBinom(r,ρ)	$m=\rho/(1-\rho)$	$(r+x)\log(1+m)-x\log(m+\epsilon)$	$x\in\mathbb{N}$, $m\geq 0$

Gradient based optimization

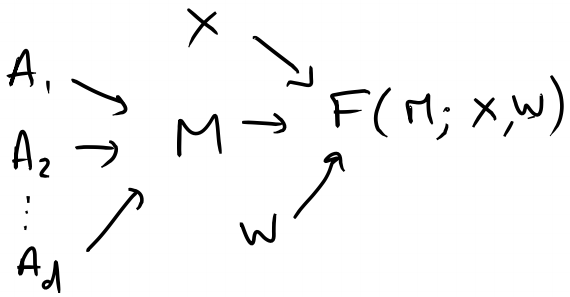
(if Ω is the set of observed entries: $W_{i_1,\dots,i_d}=1$ if $(i_1,\dots,i_d)\in\Omega$ and 0 o.w.)

$m_1\times\cdots\times m_d$

$$\star: \quad \min_{A_1,\dots,A_d} F(\mathcal{M};\mathcal{X},\mathcal{W}) \equiv \sum_{i_1,\dots,i_d} W_{i_1,\dots,i_d}\, f(x_{i_1,\dots,i_d}, m_{i_1,\dots,i_d})$$

$$\text{s.t.} \quad \mathcal{M} = [\![A_1, A_2, \dots, A_d]\!]$$

$m_1\times R \quad m_2\times R \quad m_d\times R$



We want to show

$$\frac{\partial F(\mathcal{M};\mathcal{X},\mathcal{W})}{\partial A_k} = \underset{m_k}{Y_{(k)}} \underset{R}{I} \quad \text{where } Y \in \mathbb{R}^{m_1\times\cdots\times m_d}$$

$1\times m_k\times R$

$$Y_{i_1,\dots,i_d} = W_{i_1,\dots,i_d}\, \frac{\partial f(x_{i_1,\dots,i_d}, M_{i_1,\dots,i_d})}{\partial M_{i_1,\dots,i_d}}$$

1×1

↳ of the same size as $\mathcal{X}$ but it is sparse if $\mathcal{W}$ is sparse.

$$= Y_{(k)} \left(A_d \odot \cdots \odot A_{k+1} \odot A_{k-1} \odot \cdots \odot A_1\right)$$

$$\frac{\partial F(M; X, W)}{\partial A_k} = \overbrace{\frac{\partial F(M; X, \Pi)}{\partial M}}^{y} \frac{\partial M}{\partial A_k}$$

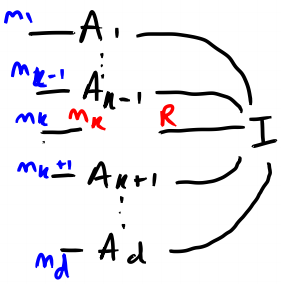
$I_X \quad m_k \times R \qquad\qquad 1 \times m_1 \times \cdots \times m_d \qquad m_1 \times \cdots \times m_d \times m_k \times R$

$$\frac{\partial F(M; X, \Pi)}{\partial M}_{i_1 \cdots i_d} = \partial \left(\sum_{j_1 \cdots j_d} W_{j_1 \cdots j_d} f(X_{j_1 \cdots j_d}, M_{j_1 \cdots j_d}) \right) / \partial M_{i_1 \cdots i_d}$$

$$= W_{i_1 \cdots i_d} \frac{\partial f(X_{i_1 \cdots i_d}, M_{i_1 \cdots i_d})}{\partial M_{i_1 \cdots i_d}} := y_{i_1 \cdots i_d}$$

$$\frac{\partial M}{\partial A_k} = \partial \left(\begin{matrix} -A_1 \\ -A_2 \\ \vdots \\ -A_d \end{matrix} \; I \right) / \partial A_k =$$

$m_1 \times \cdots \times m_d \times m_k \times R$



$$\frac{\partial F(M; X, W)}{\partial A_k} = \frac{\partial F(M; X, \Pi)}{\partial M} \frac{\partial M}{\partial A_k}$$

$$= \quad y \quad I \quad = y_{(k)} (A_d \odot \cdots \odot A_{k+1} \odot A_{k-1} \odot \cdots \odot A_1)$$

m_k R A_1 A_{k-1} A_{k+1} A_d

Follow up paper:

Stochastic Gradients for Large-Scale Tensor Decomposition*

Tamara G. Kolda† and David Hong‡