Spectral Methods II

Tensor Decompositions for Machine Learning

G. Roeder[1]

[1]Department of Computer Science
University of British Columbia

UBC Machine Learning Reading Group, June 15 2016

Contact information

Please inform me of any typos or errors at geoff.roeder@gmail.com

Outline

Outline

Formal Definition
Tensors as Multi-Linear Maps

- The tensor product of two vector spaces V and W over a field $\mathbb{F}$ is another vector space over $\mathbb{F}$. It is denoted $V \otimes_K W$ or $V \otimes W$ when the field is understood.

Formal Definition
Tensors as Multi-Linear Maps

- The tensor product of two vector spaces V and W over a field $\mathbb{F}$ is another vector space over $\mathbb{F}$. It is denoted $V \otimes_K W$ or $V \otimes W$ when the field is understood.
- If $\{v_i\}$ and $\{w_j\}$ are bases for V and W, then the set $\{v_i \otimes w_j\}$ is a basis for $V \otimes W$

Formal Definition

Tensors as Multi-Linear Maps

- The tensor product of two vector spaces V and W over a field $\mathbb{F}$ is another vector space over $\mathbb{F}$. It is denoted $V \otimes_K W$ or $V \otimes W$ when the field is understood.
- If $\{v_i\}$ and $\{w_j\}$ are bases for V and W, then the set $\{v_i \otimes w_j\}$ is a basis for $V \otimes W$
- If $S : V \to X$ and $T : W \to Y$ are linear maps, then the tensor product of S and T is a linear map

$$S \otimes T : V \otimes W \to X \otimes Y$$

defined by

$$(S \otimes T)(v \otimes w) = S(v) \otimes T(w)$$

Sufficient Definition

Tensors as Multidimensional Arrays (adapted from Kolda and Bader 2009)

- With respect to a given basis, a tensor is a multidimensional array. E.g., a real N^{th}-order tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ w.r.t. the standard basis is an N-dimensional array where $\mathcal{X}_{i_1,i_2,\ldots,i_N}$ is the element at index $(i_1, i_2, \ldots, i_N)$

Sufficient Definition

Tensors as Multidimensional Arrays (adapted from Kolda and Bader 2009)

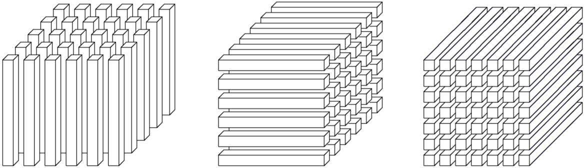
- With respect to a given basis, a tensor is a multidimensional array. E.g., a real N^{th}-order tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ w.r.t. the standard basis is an N-dimensional array where $\mathcal{X}_{i_1, i_2, \ldots, i_N}$ is the element at index $(i_1, i_2, \ldots, i_N)$
- The **order** of a tensor is the number of *dimensions* or *modes* or *indices required to uniquely identify an element*

Sufficient Definition

Tensors as Multidimensional Arrays (adapted from Kolda and Bader 2009)

- With respect to a given basis, a tensor is a multidimensional array. E.g., a real N^{th}-order tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ w.r.t. the standard basis is an N-dimensional array where $\mathcal{X}_{i_1, i_2, \ldots, i_N}$ is the element at index $(i_1, i_2, \ldots, i_N)$
- The **order** of a tensor is the number of *dimensions* or *modes* or *indices required to uniquely identify an element*
- So, a scalar is a $0-$mode tensor, a vector is a $1-$mode tensor, and a matrix is a $2-$mode tensor

- Tensor **fibers**: higher-order analogue to matrix rows and columns. For 3^{rd}-order tensor, fix all but one index.



(a) Mode-1 (column) fibers: $\mathbf{x}_{:jk}$ (b) Mode-2 (row) fibers: $\mathbf{x}_{i:k}$ (c) Mode-3 (tube) fibers: $\mathbf{x}_{ij:}$

Figure: Fibers of a 3-mode tensor

Tensors
Subarrays

- Tensor **fibers**: higher-order analogue to matrix rows and columns. For 3^{rd}-order tensor, fix all but one index.

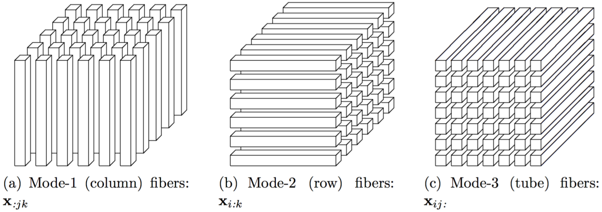
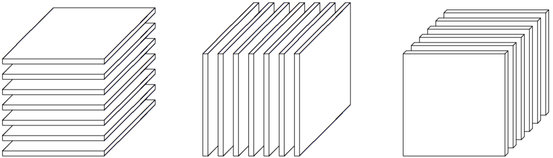


Figure: Fibers of a 3-mode tensor

- All fibers are treated as columns vectors by convention.

Tensors
Subarrays

- Tensor **slices**: two-dimensional sections of a tensor, defined by fixing all but two indices:

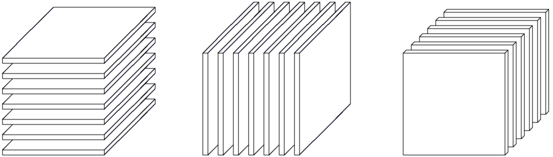


(a) Horizontal slices: $\mathbf{X}_{i::}$ (b) Lateral slices: $\mathbf{X}_{:j:}$ (c) Frontal slices: $\mathbf{X}_{::k}$ (or $\mathbf{X}_k$)

Figure: Slices of a 3^{rd}-order tensor

Tensors
Subarrays

- Tensor **slices**: two-dimensional sections of a tensor, defined by fixing all but two indices:



(a) Horizontal slices: $\mathbf{X}_{i::}$ (b) Lateral slices: $\mathbf{X}_{:j:}$ (c) Frontal slices: $\mathbf{X}_{::k}$ (or $\mathbf{X}_k$)

Figure: Slices of a 3^{rd}-order tensor

- Slices can be horizontal, lateral, or frontal, corresponding to the diagrams above.

- Consider multiplying tensor by a matrix or vector in the n^{th} mode.

- Consider multiplying tensor by a matrix or vector in the n^{th} mode.
- The **n-mode (matrix) product** of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ with a matrix $S \in \mathbb{R}^{J \times I_n}$ is denoted $\mathcal{X} \times_n S$ and lives in $\mathbb{R}^{I_1 \times \ldots \times I_{n-1} \times J \times I_{n+1} \times \ldots \times I_N}$

Tensors Operations
Tensor-Matrix Product

- Consider multiplying tensor by a matrix or vector in the n^{th} mode.
- The **n-mode (matrix) product** of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ with a matrix $S \in \mathbb{R}^{J \times I_n}$ is denoted $\mathcal{X} \times_n S$ and lives in $\mathbb{R}^{I_1 \times \ldots \times I_{n-1} \times J \times I_{n+1} \times \ldots \times I_N}$
- Elementwise,

$$(\mathcal{X} \times_n S)_{i_1, \ldots, i_{n-1}, j, i_{n+1}, \ldots, i_N} = \sum_{i_n=1}^{I_n} x_{i_1, i_2, \ldots, i_N} s_{j, i_n}$$

- **Example**. Consider the tensor $\mathcal{X} \in \mathbb{R}^{3\times2\times2}$, with frontal slices

$$\mathcal{X}_{:,:,1} = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix} \quad \mathcal{X}_{:,:,2} = \begin{bmatrix} 13 & 16 \\ 14 & 17 \\ 15 & 18 \end{bmatrix} \quad U = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 5 & 6 \end{bmatrix}$$

- **Example**. Consider the tensor $\mathcal{X} \in \mathbb{R}^{3\times2\times2}$, with frontal slices

$$\mathcal{X}_{:,:,1} = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix} \quad \mathcal{X}_{:,:,2} = \begin{bmatrix} 13 & 16 \\ 14 & 17 \\ 15 & 18 \end{bmatrix} \quad U = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 5 & 6 \end{bmatrix}$$

- Then, the $(1, 1, 1)$ element of the 1-mode matrix product of $\mathcal{X}$ and U is:

$$\begin{aligned}(\mathcal{X} \times_1 U)_{1,1,1} &= \sum_{i_1=1}^{I_1=3} x_{i_1,1,1} \times u_{1,i_1} \\ &= 1 \times 1 + 2 \times 3 + 3 \times 5 = 22\end{aligned}$$

- The **n-mode (vector) product** of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ with a vector $w \in \mathbb{R}^{I_n}$ is denoted $\mathcal{X} \bullet_n w$ and lives in $\mathbb{R}^{I_1 \times \ldots \times I_{n-1} \times I_{n+1} \times \ldots \times I_N}$.

- The **n-mode (vector) product** of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ with a vector $w \in \mathbb{R}^{I_n}$ is denoted $\mathcal{X} \bullet_n w$ and lives in $\mathbb{R}^{I_1 \times \ldots \times I_{n-1} \times I_{n+1} \times \ldots \times I_N}$.
- Elementwise,

$$(\mathcal{X} \bullet_n w)_{i_1,\ldots,i_{n-1},i_{n+1},\ldots,i_N} = \sum_{i_n=1}^{I_N} x_{i_1,i_2,\ldots,i_N} w_{i_n}$$

- **Example**. Consider the tensor $\mathcal{X} \in \mathbb{R}^{3\times2\times2}$ as defined as before, where $\mathcal{X}_{1,1,1} = 1$ and $\mathcal{X}_{1,2,1} = 4$. Then, the $(1,1)$ element of the 2-mode vector product of $\mathcal{X}$ and $w = [1\ 2]$ is

$$
\begin{aligned}
(\mathcal{X} \bullet_2 w)_{1,1} &= \sum_{i_2=1}^{I_2=2} x_{i_1,i_2,i_3} \times w_{i_2} \\
&= 1 \times 1 + 4 \times 2 = 9
\end{aligned}
$$

Multilinear form of a Tensor $\mathcal{X}$

- The multilinear form for a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ is defined as follows.

Multilinear form of a Tensor $\mathcal{X}$

- The multilinear form for a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ is defined as follows.
- Consider matrices $M_i \in \mathbb{R}^{q_i \times p_i}$ for $i \in 1, 2, 3$. Then, the tensor $\mathcal{X}(M_1, M_2, M_3) \in \mathbb{R}^{p_1 \times p_2 \times p_3}$ is defined as

$$\sum_{j_1 \in [q_1]} \sum_{j_2 \in [q_2]} \sum_{j_3 \in [q_3]} \mathcal{X}_{j_1, j_2, j_3} \cdot M_1(j_1, :) \otimes M_2(j_2, :) \otimes M_3(j_3, :)$$

Multilinear form of a Tensor $\mathcal{X}$

- The multilinear form for a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ is defined as follows.
- Consider matrices $M_i \in \mathbb{R}^{q_i \times p_i}$ for $i \in 1, 2, 3$. Then, the tensor $\mathcal{X}(M_1, M_2, M_3) \in \mathbb{R}^{p_1 \times p_2 \times p_3}$ is defined as

$$\sum_{j_1 \in [q_1]} \sum_{j_2 \in [q_2]} \sum_{j_3 \in [q_3]} \mathcal{X}_{j_1, j_2, j_3} \cdot M_1(j_1, :) \otimes M_2(j_2, :) \otimes M_3(j_3, :)$$

- A simpler case is with vectors $v, w \in \mathbb{R}^d$. Then,

$$\mathcal{X}(\mathbb{I}, v, w) := \sum_{j, l \in [d]} v_j w_l T(:, j, l) \in \mathbb{R}^d$$

which is a multilinear combination of the mode-1 fibers (columns) of the tensor $\mathcal{X}$

Rank One Tensors

- An N-way tensor $\mathcal{X}$ is called **rank one** if it can be written as the tensor product of N vectors, i.e.,

$$\mathcal{X} = a^{(1)} \otimes a^{(2)} \otimes \ldots \otimes a^{(N)}. \tag{1}$$

where $\otimes$ is the vector (outer) product. The simple form of (1) implies that

$$\mathcal{X}_{i_1, i_2, \ldots, i_N} = a^{(1)}_{i_1} \times a^{(2)}_{i_2} \times \ldots \times a^{(N)}_{i_N}$$

Rank One Tensors

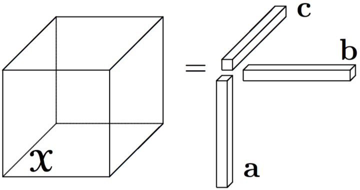
- An N-way tensor $\mathcal{X}$ is called **rank one** if it can be written as the tensor product of N vectors, i.e.,

$$\mathcal{X} = a^{(1)} \otimes a^{(2)} \otimes \ldots \otimes a^{(N)}. \qquad (1)$$

where $\otimes$ is the vector (outer) product. The simple form of (1) implies that

$$\mathcal{X}_{i_1, i_2, \ldots, i_N} = a^{(1)}_{i_1} \times a^{(2)}_{i_2} \times \ldots \times a^{(N)}_{i_N}$$

- The following diagram exhibits a **rank one** 3-mode tensor $\mathcal{X} = a \otimes b \otimes c$:



Rank k Tensors

- An order-3 tensor $\mathcal{X} \in \mathbb{R}^{d \times d \times d}$ is said to have **rank k** if it can be written as the sum of k rank-1 tensors, i.e.,

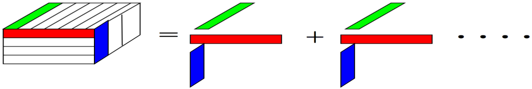
$$\mathcal{X} = \sum_{i=1}^{k} w_i \cdot a_i \otimes b_i \otimes c_i, \;\; w_i \in \mathbb{R}, \;\; a_i, b_i, c_i \in \mathbb{R}^d.$$

Rank k Tensors

- An order-3 tensor $\mathcal{X} \in \mathbb{R}^{d \times d \times d}$ is said to have **rank k** if it can be written as the sum of k rank-1 tensors, i.e.,

$$\mathcal{X} = \sum_{i=1}^{k} w_i \cdot a_i \otimes b_i \otimes c_i, \;\; w_i \in \mathbb{R}, \;\; a_i, b_i, c_i \in \mathbb{R}^d.$$

- Analogy to SVD where $M = \sum_i \sigma_i u \otimes v^\top$: suggests finding a decomposition of an arbitrary tensor into a "spectrum" of rank-one components:



Outline

Tensor Power Method
Motivation

- Consider a 3^{rd}-order tensor of the form $\mathcal{A} = \sum_i w_i a_i \otimes a_i \otimes a_i$. Considering $\mathcal{A}$ as a multilinear map, we can represent its action on lower-order input tensors (vectors and matrices) using its multilinear form:

$$\mathcal{A}(B, C, D) := \sum_i w_i (B^T a_i) \cdot (C^T a_i) \cdot (D^T a_i).$$

Tensor Power Method
Motivation

- Consider a 3^{rd}-order tensor of the form $\mathcal{A} = \sum_i w_i a_i \otimes a_i \otimes a_i$. Considering $\mathcal{A}$ as a multilinear map, we can represent its action on lower-order input tensors (vectors and matrices) using its multilinear form:

$$\mathcal{A}(B, C, D) := \sum_i w_i (B^T a_i) \cdot (C^T a_i) \cdot (D^T a_i).$$

- Now suppose $\mathcal{A}$ had orthonormal columns. Then, $M_3(\mathbb{I}, a_1, a_1) = \sum_i w_i \cdot (\mathbb{I}^\top a_i) \cdot (a_i^\top a_1)^2 = w_1 a_1 + 0 + 0.$

- Consider a 3^{rd}-order tensor of the form $\mathcal{A} = \sum_i w_i a_i \otimes a_i \otimes a_i$. Considering $\mathcal{A}$ as a multilinear map, we can represent its action on lower-order input tensors (vectors and matrices) using its multilinear form:

$$\mathcal{A}(B, C, D) := \sum_i w_i (B^T a_i) \cdot (C^T a_i) \cdot (D^T a_i).$$

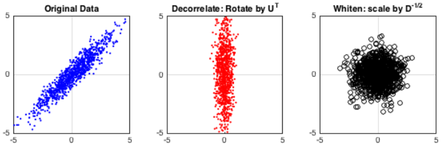
- Now suppose $\mathcal{A}$ had orthonormal columns. Then, $M_3(\mathbb{I}, a_1, a_1) = \sum_i w_i \cdot (\mathbb{I}^\top a_i) \cdot (a_i^\top a_1)^2 = w_1 a_1 + 0 + 0.$
- This is analogous to an eigenvector of a matrix. If v is an eigenvector of M we can write

$$Mv = M(\mathbb{I}, v) = \lambda v$$

- We're extremely unlikely to encounter an empirical tensor built from orthogonal components like this...

- We're extremely unlikely to encounter an empirical tensor built from orthogonal components like this...
- But can learn a whitening transform using the second moment, $\mathcal{M}_2 = \sum_i w_i a_i \otimes a_i \equiv \sum_i w_i a_i a_i^\top$.

- We're extremely unlikely to encounter an empirical tensor built from orthogonal components like this...
- But can learn a whitening transform using the second moment, $\mathcal{M}_2 = \sum_i w_i a_i \otimes a_i \equiv \sum_i w_i a_i a_i^\top$.
- Whitening transforms the covariance matrix to the identity matrix. The data is thereby *decorrelated* with *unit variance*. The following diagram displays the action of a whitening transform on data sampled from a bivariate Gaussian:

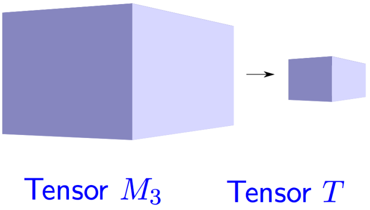


- The whitening transform is invertible so long as the empirical second moment matrix has full column rank

- The whitening transform is invertible so long as the empirical second moment matrix has full column rank
- Given the whitening matrix W, we can whiten the empirical third moment tensor by evaluating

$$\mathcal{T} = \mathcal{M}_3(W, W, W) = \sum_i w_i \cdot (W^\top a_i)^{\otimes 3} = \sum_{i \in [k]} w_i \cdot v_i^{\otimes 3}$$

where $\{v_i\}$ is now an orthogonal basis



Tensor Power Method

Procedure

- Start from a whitened tensor $\mathcal{T}$. Then:

Tensor Power Method

Procedure

- Start from a whitened tensor $\mathcal{T}$. Then:
  1. Randomly initialize v. Evaluate the expression

$$v \mapsto \frac{\mathcal{T}(\mathbb{I}, v, v)}{||\mathcal{T}(\mathbb{I}, v, v,)||}$$

     until convergence to obtain v with eigenvalue λ

Tensor Power Method

Procedure

- Start from a whitened tensor $\mathcal{T}$. Then:
  1. Randomly initialize v. Evaluate the expression
  $$v \mapsto \frac{\mathcal{T}(\mathbb{I}, v, v)}{||\mathcal{T}(\mathbb{I}, v, v,)||}$$
  until convergence to obtain v with eigenvalue λ
  2. Deflate $\mathcal{T} = \mathcal{T} - \lambda v \otimes v \otimes v$. Store eigenvalue/eigenvector pair, and then go to 1.

Tensor Power Method

Procedure

- Start from a whitened tensor $\mathcal{T}$. Then:
  1. Randomly initialize v. Evaluate the expression
  $$v \mapsto \frac{\mathcal{T}(\mathbb{I}, v, v)}{||\mathcal{T}(\mathbb{I}, v, v,)||}$$
  until convergence to obtain v with eigenvalue λ
  2. Deflate $\mathcal{T} = \mathcal{T} - \lambda v \otimes v \otimes v$. Store eigenvalue/eigenvector pair, and then go to 1.
- This leads to the algorithm for recovering the columns of a parameter matrix by representing its columns as moments:

Input: Tensor $T = \sum_{i \in [k]} \lambda_i u_i^{\otimes 3}$

$\downarrow$

Whitening procedure (Procedure 5)

$\downarrow$

SVD-based Initialization (Procedure 8)

$\downarrow$

Tensor Power Method (Algorithm 7)

$\downarrow$

Output: $\{u_i\}_{i \in [k]}$

Summary of Method of Moments

- Tensor factorization is NP-hard in general

Summary of Method of Moments

- Tensor factorization is NP-hard in general
- For orthogonal tensors, factorization is *polynomial* in sample size and number of operations

Summary of Method of Moments

- Tensor factorization is NP-hard in general
- For orthogonal tensors, factorization is *polynomial* in sample size and number of operations
- Unlike EM algorithm or variational Bayes, this method converges to the global optimum

Summary of Method of Moments

- Tensor factorization is NP-hard in general
- For orthogonal tensors, factorization is *polynomial* in sample size and number of operations
- Unlike EM algorithm or variational Bayes, this method converges to the global optimum
- For a more detailed analysis and how to frame any latent variable model using this method, see `newport.eecs.uci.edu/anandkumar/MLSS.html`

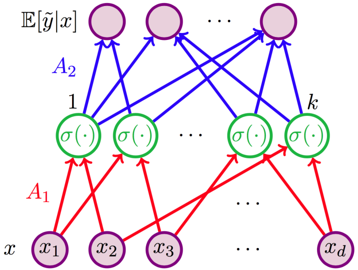
Outline

- Very recent work (last arXiv.org datestamp: Jan 11 2016) on finding optimal weights for a two-layer neural network, with notes on how to generalize to more complex architectures: `https://arxiv.org/pdf/1506.08473.pdf`

- Very recent work (last arXiv.org datestamp: Jan 11 2016) on finding optimal weights for a two-layer neural network, with notes on how to generalize to more complex architectures: `https://arxiv.org/pdf/1506.08473.pdf`
- Majid Janzamin, Hanie Sedghi, and Anima Anandkumar. *Beating the Perils of Non-Convexity: Guaranteed Training of Neural Networks using Tensor Methods.*

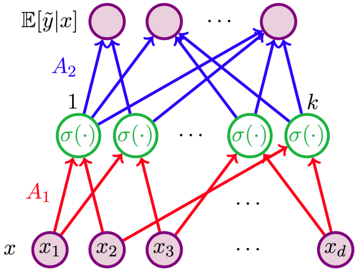
Neural Network LearnIng Using Feature Tensors (NNLIFT)

- Target network is a label-generating model with architecture $\tilde{f}(x) := \mathbb{E}[\tilde{y}|x] = A_1^\top \sigma(A_2^\top x + b_1) + b_2$:



Neural Network LearnIng Using Feature Tensors (NNLIFT)

- Target network is a label-generating model with architecture $\tilde{f}(x) := \mathbb{E}[\tilde{y}|x] = A_1^\top \sigma(A_2^\top x + b_1) + b_2$:



- Must assume input pdf $p(x)$ is known "sufficiently well" for learning (or can be estimated using unsupervised methods)

Neural Network LearnIng Using Feature Tensors (NNLIFT)

- Key insight: there exists a transformation $\phi(\cdot)$ of the input $\{(x_i, y_i)\}$ that captures the relationship between the parameter matrices A_1 and A_2 and the input

Neural Network LearnIng Using Feature Tensors (NNLIFT)

- Key insight: there exists a transformation $\phi(\cdot)$ of the input $\{(x_i, y_i)\}$ that captures the relationship between the parameter matrices A_1 and A_2 and the input
- The transformation generates feature tensors that can be factorized using the method of moments

Neural Network LearnIng Using Feature Tensors (NNLIFT)

- Key insight: there exists a transformation $\phi(\cdot)$ of the input $\{(x_i, y_i)\}$ that captures the relationship between the parameter matrices A_1 and A_2 and the input
- The transformation generates feature tensors that can be factorized using the method of moments
- m^{th}**-order Score function**, defined as (Janzamin et al. 2014)

$$S_m(x) := (-1)^m \frac{\nabla_x^{(m)} p(x)}{p(x)}$$

where $p(x)$ is the pdf of the random vector $x \in \mathbb{R}^d$ and $\nabla_x^{(m)}$ is the m^{th} order derivative operator

- The 1st order score function is the normalized gradient of the log of the input density function

- The 1st order score function is the normalized gradient of the log of the input density function
- This encodes variations in the input distribution $p(x)$. By looking at the gradient of the distribution you get an idea of where there is a large change occurring in the distribution

- The 1st order score function is the normalized gradient of the log of the input density function
- This encodes variations in the input distribution $p(x)$. By looking at the gradient of the distribution you get an idea of where there is a large change occurring in the distribution
- The correlation $\mathbb{E}[y \cdot S_3(x)]$ between the third-order score function $S_3(x)$ and the output y then has a particularly useful form, because the x averages out in expectation.

Neural Network LearnIng Using Feature Tensors (NNLIFT)

- In Lemma 6 of the paper, authors prove that the rank-1 components of the third order tensor $\mathbb{E}[\tilde{y} \cdot \mathcal{S}_3(x)]$ are the columns of the weight matrix A_1:

$$\mathbb{E}[\tilde{y} \cdot \mathcal{S}_3(x)] = \sum_{j=1}^{k} \lambda_j \cdot (A_1)_j \otimes (A_1)_j \otimes (A_1)_j \in \mathbb{R}^{d \times d \times d}$$

Neural Network LearnIng Using Feature Tensors (NNLIFT)

- In Lemma 6 of the paper, authors prove that the rank-1 components of the third order tensor $\mathbb{E}[\tilde{y} \cdot \mathcal{S}_3(x)]$ are the columns of the weight matrix A_1:

$$\mathbb{E}[\tilde{y} \cdot \mathcal{S}_3(x)] = \sum_{j=1}^{k} \lambda_j \cdot (A_1)_j \otimes (A_1)_j \otimes (A_1)_j \in \mathbb{R}^{d \times d \times d}$$

- It follows that the columns of A_1 are recoverable using the method of moments, with optimal convergence guarantees

Neural Network LearnIng Using Feature Tensors (NNLIFT)

The remainder of the steps can to the following algorithm:

Algorithm 1 NN-LIFT (Neural Network LearnIng using Feature Tensors)

input Labeled samples $\{(x_i, y_i) : i \in [n]\}$, parameter $\tilde{\epsilon}_1$, parameter λ.
input Third order score function $\mathcal{S}_3(x)$ of the input x; see Equation (8) for the definition.

1: Compute $\widehat{T} := \frac{1}{n} \sum_{i \in [n]} y_i \cdot \mathcal{S}_3(x_i)$.
2: $\{(\hat{A}_1)_j\}_{j \in [k]} =$ tensor decomposition$(\widehat{T})$; see Section 3.2 and Appendix B for details.
3: $\hat{b}_1 =$ Fourier method$(\{(x_i, y_i) : i \in [n]\}, \hat{A}_1, \tilde{\epsilon}_1)$; see Procedure 2.
4: $(\hat{a}_2, \hat{b}_2) =$ Ridge regression$(\{(x_i, y_i) : i \in [n]\}, \hat{A}_1, \hat{b}_1, \lambda)$; see Procedure 3.
5: **return** $\hat{A}_1, \hat{a}_2, \hat{b}_1, \hat{b}_2$.

Summary of Talk

- Tensorial representations of Latent Variable Models promise to overcome shortcomings of EM algorithm and variational Bayes

Summary of Talk

- Tensorial representations of Latent Variable Models promise to overcome shortcomings of EM algorithm and variational Bayes
- Tensors algebra involves non-trivial but conceptually straightforward operations

Summary of Talk

- Tensorial representations of Latent Variable Models promise to overcome shortcomings of EM algorithm and variational Bayes
- Tensors algebra involves non-trivial but conceptually straightforward operations
- These methods may point to a new direction in machine learning research that gives guarantees in unsupervised learning

References I

- Anandkumar, Anima. Tutorial on Spectral and Tensor Methods for Guaranteed Learning, Part 1. (2014). *Lecture slides, Machine Learning Summer School 2014, Carnegie Mellon University* http://newport.eecs.uci.edu/anandkumar/MLSS.html

- Janzamin, Majid, Hanie Sedghi, and Anima Anandkumar. Beating the Perils of Non-Convexity: Guaranteed Training of Neural Networks using Tensor Methods. *arXiv:1506.08473v3 [cs.LG]* http://arxiv.org/abs/1506.08473v3

- Kolda, Tamara G. Brett W. Bader. "Tensor Decompositions and Applications." *SIAM Review, 51(3), pp.455-500*