Guaranteed Machine Learning using Tensor Methods

Anima Anandkumar

U.C. Irvine

Tutorial at MIT 2016
Regime of Modern Machine Learning
Massive datasets, growth in computation power, challenging tasks

Success of Supervised Learning

- Learn $p(y|x)$ from labeled samples $\{(x_i, y_i)\}$.
- Extract relevant features from large amounts of labeled data.

Image classification | Speech recognition | Text processing
Regime of Modern Machine Learning

Massive datasets, growth in computation power, challenging tasks

Missing Link in AI: Unsupervised Learning

- Learn $p(x)$ from unlabeled samples $\{x_i\}$.
- Discover latent variables related to observed variable $x$.
Unsupervised Learning via Probabilistic Models

Data $\rightarrow$ Model $\rightarrow$ Learning Algorithm $\rightarrow$ Predictions

Challenges in High dimensional Learning

- Dimension of $x \gg$ dim. of latent variable $h$.
- Learning is like finding needle in a haystack.
- Computationally & statistically challenging.
Overview of Unsupervised Learning Methods

Goal: learn model parameters $\theta$ from observations $x$.

Bayesian: treat $\theta$ as random
- Compute the posterior distribution $p(\theta|x)$.
- Algorithm: Markov chain Monte Carlo (MCMC).
- Slow: Exponential mixing time.

Frequentist: treat $\theta$ as fixed
- Maximum likelihood: $\max_\theta p(x; \theta)$.
- Non-convex: stuck in local optima.
- Curse of dimensionality: Exponential no. of critical points.
- Heuristics: Expectation Maximization, Variational Inference . . .
Guaranteed Learning through Tensor Methods

Replace the objective function
Max Likelihood vs. Best Tensor decomp.

Preserves Global Optimum (infinite samples)

\[
\arg \max_{\theta} p(x; \theta) = \arg \min_{\theta} \| \hat{T}(x) - T(\theta) \|_F^2
\]

\( \hat{T}(x) \): empirical tensor, \( T(\theta) \): low rank tensor based on \( \theta \).
Guaranteed Learning through Tensor Methods

Replace the objective function
Max Likelihood vs. Best Tensor decomp.

Preserves Global Optimum (infinite samples)

\[
\arg\max_{\theta} p(x; \theta) = \arg\min_{\theta} \|\hat{T}(x) - T(\theta)\|_F^2
\]

\(\hat{T}(x)\): empirical tensor, \(T(\theta)\): low rank tensor based on \(\theta\).

Finding globally opt tensor decomposition
Simple algorithms succeed under mild and natural conditions for many learning problems.
Guaranteed Learning through Tensor Methods

Replace the objective function
Max Likelihood vs. Best Tensor decomp.

Preserves Global Optimum (infinite samples)

$$\arg \max_{\theta} p(x; \theta) = \arg \min_{\theta} \| \hat{T}(x) - T(\theta) \|_F^2$$

$\hat{T}(x)$: empirical tensor, $T(\theta)$: low rank tensor based on $\theta$.

Finding globally opt tensor decomposition
Simple algorithms succeed under mild and natural conditions for many learning problems.
Outline

1. Introduction
2. Why Tensors?
3. Algorithms for Tensor Decomposition
4. Learning Probabilistic Models
5. Learning Representations
6. Conclusion
Recall PCA

Optimization problem

For (centered) points $x_i \in \mathbb{R}^d$, find projection $P$ with $\text{Rank}(P) = k$ s.t.

$$\min_{P \in \mathbb{R}^{d \times d}} \frac{1}{n} \sum_{i \in [n]} \|x_i - Px_i\|^2.$$
Optimization problem

For (centered) points $x_i \in \mathbb{R}^d$, find projection $P$ with $\text{Rank}(P) = k$ s.t.

$$\min_{P \in \mathbb{R}^{d \times d}} \frac{1}{n} \sum_{i \in [n]} \|x_i - Px_i\|^2.$$ 

Result: If $X = [x_1 | x_2 \ldots]$ is data matrix, $S = \text{Cov}(X)$, and $S = U \Lambda U^\top$ is eigen decomposition, we have $P = U(k) U(k)^\top$, where $U(k)$ are top-$k$ eigen vectors.
PCA on Gaussian Mixtures

- $k$ Gaussians: each sample is $x = Ah + z$.
- $h \in [e_1, \ldots, e_k]$, the basis vectors. $\mathbb{E}[h] = w$.
- $A \in \mathbb{R}^{d \times k}$: columns are component means.
- Let $\mu := Aw$ be the mean.
- $z \sim \mathcal{N}(0, \sigma^2 I)$ is white Gaussian noise.
PCA on Gaussian Mixtures

- $k$ Gaussians: each sample is $x = Ah + z$.
- $h \in [e_1, \ldots, e_k]$, the basis vectors. $\mathbb{E}[h] = w$.
- $A \in \mathbb{R}^{d \times k}$: columns are component means.
- Let $\mu := Aw$ be the mean.
- $z \sim \mathcal{N}(0, \sigma^2 I)$ is white Gaussian noise.

$$\mathbb{E}[(x - \mu)(x - \mu)^\top] = \sum_{i \in [k]} w_i (a_i - \mu)(a_i - \mu)^\top + \sigma^2 I.$$
PCA on Gaussian Mixtures

- \( k \) Gaussians: each sample is \( x = Ah + z \).
- \( h \in [e_1, \ldots, e_k] \), the basis vectors. \( \mathbb{E}[h] = w \).
- \( A \in \mathbb{R}^{d \times k} \): columns are component means.
- Let \( \mu := Aw \) be the mean.
- \( z \sim \mathcal{N}(0, \sigma^2 I) \) is white Gaussian noise.

\[
\mathbb{E}[(x - \mu)(x - \mu)^\top] = \sum_{i \in [k]} w_i(a_i - \mu)(a_i - \mu)^\top + \sigma^2 I.
\]

How the above equation is obtained

\[
\mathbb{E}[(x - \mu)(x - \mu)^\top] = \mathbb{E}[(Ah - \mu)(Ah - \mu)^\top] + \mathbb{E}[zz^\top]
= \sum_{i \in [k]} w_i(a_i - \mu)(a_i - \mu)^\top + \sigma^2 I.
\]
Example 1: PCA on Gaussian Mixtures

\[
\mathbb{E}[(x - \mu)(x - \mu)^\top] = \sum_{i \in [k]} w_i (a_i - \mu)(a_i - \mu)^\top + \sigma^2 I.
\]

- The vectors \( \{a_i - \mu\} \) are linearly dependent: \( \sum_i w_i (a_i - \mu) = 0 \). The PSD matrix \( \sum_{i \in [k]} w_i (a_i - \mu)(a_i - \mu)^\top \) has rank \( \leq k - 1 \).
Example 1: PCA on Gaussian Mixtures

\[
\mathbb{E}[(x - \mu)(x - \mu)^\top] = \sum_{i \in [k]} w_i (a_i - \mu)(a_i - \mu)^\top + \sigma^2 I.
\]

- The vectors \( \{a_i - \mu\} \) are linearly dependent: \( \sum_i w_i (a_i - \mu) = 0 \). The PSD matrix \( \sum_{i \in [k]} w_i (a_i - \mu)(a_i - \mu)^\top \) has rank \( \leq k - 1 \).
- \((k - 1)\)-PCA on covariance matrix \( \cup \{\mu\} \) yields \( \text{span}(A) \).
- Lowest eigenvalue of covariance matrix yields \( \sigma^2 \).
Example 1: PCA on Gaussian Mixtures

\[
\mathbb{E}[(x - \mu)(x - \mu)^\top] = \sum_{i \in [k]} w_i (a_i - \mu)(a_i - \mu)^\top + \sigma^2 I.
\]

- The vectors \( \{a_i - \mu\} \) are linearly dependent: \( \sum_i w_i (a_i - \mu) = 0 \). The PSD matrix \( \sum_{i \in [k]} w_i (a_i - \mu)(a_i - \mu)^\top \) has rank \( \leq k - 1 \).

- \((k - 1)\)-PCA on covariance matrix \( \cup \{\mu\} \) yields \( \text{span}(A) \).

- Lowest eigenvalue of covariance matrix yields \( \sigma^2 \).

How to Learn \( A \)?
Learning through Spectral Clustering

Learning $A$ through Spectral Clustering

- Project samples $x$ on to span$(A)$.
- Distance-based clustering (e.g. $k$-means).
- A series of works, e.g. Vempala & Wang.
Learning through Spectral Clustering

Learning $A$ through Spectral Clustering

- Project samples $x$ on to $\text{span}(A)$.
- Distance-based clustering (e.g. $k$-means).
- A series of works, e.g. Vempala & Wang.

Failure to cluster under large variance.

Learning Gaussian Mixtures Without Separation Constraints?
Matrix: Pairwise Moments

- $\mathbb{E}[x \otimes x] \in \mathbb{R}^{d \times d}$ is a second order tensor.
- $\mathbb{E}[x \otimes x]_{i_1, i_2} = \mathbb{E}[x_{i_1} x_{i_2}]$.
- For matrices: $\mathbb{E}[x \otimes x] = \mathbb{E}[xx^\top]$.
- $M = uu^\top$ is rank-1 and $M_{i,j} = u_i u_j$.

Tensor: Higher order Moments

- $\mathbb{E}[x \otimes x \otimes x] \in \mathbb{R}^{d \times d \times d}$ is a third order tensor.
- $\mathbb{E}[x \otimes x \otimes x]_{i_1, i_2, i_3} = \mathbb{E}[x_{i_1} x_{i_2} x_{i_3}]$.
- $T = u \otimes u \otimes u$ is rank-1 and $T_{i,j,k} = u_i u_j u_k$. 
Third order moment for Gaussian mixtures

\[ \mathbb{E}[x \otimes x \otimes x] = \sum_i w_i a_i \otimes a_i \otimes a_i + \sigma^2 \sum_i (\mu \otimes e_i \otimes e_i + \ldots) \]

- \( \sigma^2 \) is obtained from \( \sigma_{\text{min}}(\text{Cov}(x)) \).
- \( \mathbb{E}[x \otimes x] = \sum_i w_i a_i \otimes a_i + \sigma^2 I. \)
Third order moment for Gaussian mixtures

\[ \mathbb{E}[x \otimes x \otimes x] = \sum_i w_i a_i \otimes a_i \otimes a_i + \sigma^2 \sum_i (\mu \otimes e_i \otimes e_i + \ldots) \]

- \( \sigma^2 \) is obtained from \( \sigma_{\text{min}}(\text{Cov}(x)) \).
- \( \mathbb{E}[x \otimes x] = \sum_i w_i a_i \otimes a_i + \sigma^2 I. \)

Can obtain

\[ M_3 = \sum_i w_i a_i \otimes a_i \otimes a_i \]
\[ M_2 = \sum_i w_i a_i \otimes a_i. \]

Obtain parameters \( A \) and \( w \) from \( M_3 \).
Visualization
Example 2: Discovering Latent Factors

- List of scores for students in different tests
- Learn hidden factors for Verbal and Mathematical Intelligence [C. Spearman 1904]

\[
\text{Score} \ (\text{student}, \text{test}) = \text{student}_{\text{verbal-intlg}} \times \text{test}_{\text{verbal}} + \text{student}_{\text{math-intlg}} \times \text{test}_{\text{math}}
\]
Matrix Decomposition: Discovering Latent Factors

- Identifying **hidden factors** influencing the observations
- Characterized as **matrix decomposition**
Matrix Decomposition: Discovering Latent Factors

- Decomposition is **not necessarily unique**.
- Decomposition cannot be **overcomplete**.
Tensor: Shared Matrix Decomposition

- Shared decomposition with different scaling factors
- Combine matrix slices as a tensor
Tensor Decomposition

- Outer product notation:

\[
T = u \otimes v \otimes w + \tilde{u} \otimes \tilde{v} \otimes \tilde{w}
\]

\[
T_{i_1,i_2,i_3} = u_{i_1} \cdot v_{i_2} \cdot w_{i_3} + \tilde{u}_{i_1} \cdot \tilde{v}_{i_2} \cdot \tilde{w}_{i_3}
\]
Tensor Decomposition

Uniqueness of Tensor Decomposition [J. Kruskal 1977]

- Above tensor decomposition: unique when rank one pairs are linearly independent
- Matrix case: when rank one pairs are orthogonal
Example 3: Learning Naive Bayes Model

- Naive Bayes Model: $x_1 \perp x_2 \perp x_3|h$,
- $h = e_i$ for state $i$, $\mathbb{E}[x_i|h] = Ah$, $\mathbb{E}[h] = \lambda$.
- Learn matrix $A = [a_1|a_2\ldots a_k]$. 

![Diagram](image)
Example 3: Learning Naive Bayes Model

- Naive Bayes Model: \( x_1 \perp \perp x_2 \perp \perp x_3|h, \)
- \( h = e_i \) for state \( i \), \( \mathbb{E}[x_i|h] = Ah, \mathbb{E}[h] = \lambda. \)
- Learn matrix \( A = [a_1|a_2 \ldots a_k]. \)

Learning using matrix methods

- Pairwise moment \( M_2 := \mathbb{E}[x_1 x_2^\top] = \mathbb{E}[\mathbb{E}[x_1 x_2^\top|h]] = \sum_i \lambda_i a_i a_i^\top \)
- Given matrix \( M_2 \), recover \( A \)?
Example 3: Learning Naive Bayes Model

- Naive Bayes Model: \( x_1 \perp x_2 \perp x_3 | h, \)
- \( h = e_i \) for state \( i \), \( \mathbb{E}[x_i | h] = Ah, \mathbb{E}[h] = \lambda. \)
- Learn matrix \( A = [a_1 | a_2 \ldots a_k]. \)

Learning using matrix methods

- Pairwise moment \( M_2 := \mathbb{E}[x_1 x_2^\top] = \mathbb{E}[\mathbb{E}[x_1 x_2^\top | h]] = \sum_i \lambda_i a_i a_i^\top \)
- Given matrix \( M_2 \), recover \( A \)?

Matrix decomposition unique only for orthogonal factors.
Example 3: Learning Naive Bayes Model

- Naive Bayes Model: \( x_1 \perp \perp x_2 \perp \perp x_3 | h, \)
- \( h = e_i \) for state \( i, \mathbb{E}[x_i|h] = Ah, \mathbb{E}[h] = \lambda. \)
- Learn matrix \( A = [a_1 a_2 \ldots a_k]. \)

Learning using matrix methods

- Pairwise moment \( M_2 := \mathbb{E}[x_1 x_2^\top] = \mathbb{E}[\mathbb{E}[x_1 x_2^\top|h]] = \sum_i \lambda_i a_i a_i^\top \)
- Given matrix \( M_2, \) recover \( A? \)

Matrix decomposition unique only for orthogonal factors.
Example 3: Learning Naive Bayes Model

- Naive Bayes Model: \( x_1 \perp x_2 \perp x_3 | h, \)
- \( h = e_i \) for state \( i, \) \( \mathbb{E}[x_i | h] = Ah, \) \( \mathbb{E}[h] = \lambda. \)
- Learn matrix \( A = [a_1 | a_2 \ldots a_k]. \)

Learning using matrix methods

- Pairwise moment \( M_2 := \mathbb{E}[x_1 x_2^\top] = \mathbb{E}[\mathbb{E}[x_1 x_2^\top | h]] = \sum_i \lambda_i a_i a_i^\top \)
- Given matrix \( M_2, \) recover \( A? \)

Matrix decomposition unique only for orthogonal factors.
Moments of Naive Bayes Model

- \( M_2 = \mathbb{E}[x_1 \otimes x_2] = \sum_i \lambda_i a_i \otimes a_i \).
- \( M_3 = \mathbb{E}[x_1 \otimes x_2 \otimes x_3] = \sum_i \lambda_i a_i \otimes a_i \otimes a_i \).

Recover \( A = [a_1 | a_2 \ldots a_k] \) from decomposing \( M_3 \)
Outline

1. Introduction
2. Why Tensors?
3. Algorithms for Tensor Decomposition
4. Learning Probabilistic Models
5. Learning Representations
6. Conclusion
Matrix Eigen-analysis
Find decomposition of matrix $M = \sum \lambda_i v_i v_i^\top$.

- Optimization: find top eigenvector.

$$\max_v \langle v, Mv \rangle \text{ s.t. } ||v|| = 1, v \in \mathbb{R}^d.$$
Matrix Eigen-analysis
Find decomposition of matrix $M = \sum_i \lambda_i v_i v_i^\top$.

- **Optimization:** find top eigenvector.
  $$\max_v \langle v, Mv \rangle \text{ s.t. } ||v|| = 1, v \in \mathbb{R}^d.$$

- **Lagrangian:**
  $$\langle v, Mv \rangle - \lambda(||v||^2 - 1).$$
Matrix Eigen-analysis

Find decomposition of matrix \( M = \sum_i \lambda_i v_i v_i^\top. \)

- **Optimization:** find top eigenvector.
  \[
  \max_v \langle v, Mv \rangle \text{ s.t. } \|v\| = 1, v \in \mathbb{R}^d.
  \]

- **Lagrangian:**
  \[
  \langle v, Mv \rangle - \lambda (\|v\|^2 - 1).
  \]

- **Critical points:** \( Mv = \lambda v \) (all eigenvectors).
Matrix Eigen-analysis

Find decomposition of matrix $M = \sum_i \lambda_i v_i v_i^T$.

- **Optimization**: find top eigenvector.

\[
\max_v \langle v, M v \rangle \quad \text{s.t.} \quad \|v\| = 1, \quad v \in \mathbb{R}^d.
\]

- **Lagrangian**: \[
\langle v, M v \rangle - \lambda (\|v\|^2 - 1).
\]

- **Critical points**: $M v = \lambda v$ (all eigenvectors).

- All saddle points (at most $d$) are **non-degenerate**.
Matrix Eigen-analysis

Find decomposition of matrix $M = \sum_i \lambda_i v_i v_i^\top$.

- **Optimization:** find top eigenvector.
  
  $$\max_v \langle v, Mv \rangle \text{ s.t. } \|v\| = 1, v \in \mathbb{R}^d.$$ 

- **Lagrangian:**
  
  $$\langle v, Mv \rangle - \lambda (\|v\|^2 - 1).$$ 

- **Critical points:** $Mv = \lambda v$ (all eigenvectors).

- **All saddle points** (at most $d$) are non-degenerate.

- **No. of local optima:** 1
Matrix Eigen-analysis
Find decomposition of matrix \( M = \sum_i \lambda_i v_i v_i^\top \).

- **Optimization:** find top eigenvector.
  \[
  \max_v \langle v, Mv \rangle \text{ s.t. } ||v|| = 1, v \in \mathbb{R}^d.
  \]

- **Lagrangian:** \( \langle v, Mv \rangle - \lambda (||v||^2 - 1) \).

- **Critical points:** \( Mv = \lambda v \) (all eigenvectors).
- All saddle points (at most \( d \)) are **non-degenerate**.
- No. of local optima: 1

Algorithmic implication
- Gradient ascent (power method) converges to global optimum!
- Saddle points avoided by random initialization!
Matrix Eigen-analysis
Find decomposition of matrix $M = \sum_i \lambda_i v_i v_i^\top$.

- Optimization: find top eigenvector.
  \[
  \max_v \langle v, Mv \rangle \quad \text{s.t.} \quad \|v\| = 1, \ v \in \mathbb{R}^d.
  \]

- Lagrangian: $\langle v, Mv \rangle - \lambda (\|v\|^2 - 1)$.
- Critical points: $Mv = \lambda v$ (all eigenvectors).
- All saddle points (at most $d$) are non-degenerate.
- No. of local optima: 1

Local optimum $\equiv$ Global optimum!

Algorithmic implication
- Gradient ascent (power method) converges to global optimum!
- Saddle points avoided by random initialization!
Matrix: Pairwise Moments

- $\mathbb{E}[x \otimes x] \in \mathbb{R}^{d \times d}$ is a second order tensor.
- $\mathbb{E}[x \otimes x]_{i_1,i_2} = \mathbb{E}[x_{i_1} x_{i_2}]$.
- For matrices: $\mathbb{E}[x \otimes x] = \mathbb{E}[xx^\top]$.
- $M = uu^\top$ is rank-1 and $M_{i,j} = u_i u_j$.

Tensor: Higher order Moments

- $\mathbb{E}[x \otimes x \otimes x] \in \mathbb{R}^{d \times d \times d}$ is a third order tensor.
- $\mathbb{E}[x \otimes x \otimes x]_{i_1,i_2,i_3} = \mathbb{E}[x_{i_1} x_{i_2} x_{i_3}]$.
- $T = u \otimes u \otimes u$ is rank-1 and $T_{i,j,k} = u_i u_j u_k$. 
Notion of Tensor Contraction

Extends the notion of matrix product

**Matrix product**

\[ Mv = \sum_j v_j M_j \]

**Tensor Contraction**

\[ T(u, v, \cdot) = \sum_{i,j} u_i v_j T_{i,j,:} \]
Problem of Tensor Decomposition

- Computationally hard for general tensors.
- Orthogonal tensors $T = \sum_{i \in [k]} \lambda_i u_i \otimes u_i \otimes u_i$ \iff $u_i \perp u_j$ for $i \neq j$.

Decomposition through Tensor Norm Max.

$$\max_{v} T(v, v, v), \|v\| = 1, v \in \mathbb{R}^d.$$  

- Lagrangian: $T(v, v, v) - 1.5\lambda(\|v\|^2 - 1)$. 
Problem of Tensor Decomposition

- Computationally hard for general tensors.
- Orthogonal tensors $T = \sum_{i \in [k]} \lambda_i u_i \otimes u_i \otimes u_i$: $u_i \perp u_j$ for $i \neq j$.

Decomposition through Tensor Norm Max.

$$\max_v T(v, v, v), \|v\| = 1, v \in \mathbb{R}^d.$$

- Lagrangian: $T(v, v, v) - 1.5\lambda (\|v\|^2 - 1)$.
- Critical points: $T(v, v, \cdot) = \lambda v$. 
Problem of Tensor Decomposition

- Computationally hard for general tensors.
- Orthogonal tensors \( T = \sum_{i \in [k]} \lambda_i u_i \otimes u_i \otimes u_i \): \( u_i \perp u_j \) for \( i \neq j \).

Decomposition through Tensor Norm Max.

\[
\max_v T(v, v, v), \quad \|v\| = 1, \quad v \in \mathbb{R}^d.
\]

- Lagrangian: \( T(v, v, v) - 1.5\lambda(\|v\|^2 - 1) \).
- Critical points: \( T(v, v, \cdot) = \lambda v \).
- No. of eigenvectors: \( \exp(d) \)!
Problem of Tensor Decomposition

- Computationally hard for general tensors.
- Orthogonal tensors
  \[ T = \sum_{i \in [k]} \lambda_i u_i \otimes u_i \otimes u_i : u_i \perp u_j \text{ for } i \neq j. \]

Decomposition through Tensor Norm Max.

\[ \max_v T(v, v, v), \|v\| = 1, v \in \mathbb{R}^d. \]

- Lagrangian:
  \[ T(v, v, v) - 1.5\lambda(\|v\|^2 - 1). \]
- Critical points: \( T(v, v, \cdot) = \lambda v. \)
- No. of eigenvectors: \( \exp(d)! \)
- All saddle points are non-degenerate.
Problem of Tensor Decomposition

- Computationally hard for general tensors.
- Orthogonal tensors $T = \sum_{i \in [k]} \lambda_i u_i \otimes u_i \otimes u_i$: $u_i \perp u_j$ for $i \neq j$.

Decomposition through Tensor Norm Max.

$$\max_v T(v, v, v), \|v\| = 1, v \in \mathbb{R}^d.$$  

- Lagrangian: $T(v, v, v) - 1.5\lambda(\|v\|^2 - 1)$.
- Critical points: $T(v, v, \cdot) = \lambda v$.
- No. of eigenvectors: $\exp(d)!$
- All saddle points are non-degenerate.
- Local optima: $\{u_i\}$ for $i = 1, \ldots, k$. 
Problem of Tensor Decomposition

- Computationally hard for general tensors.
- Orthogonal tensors \( T = \sum_{i \in [k]} \lambda_i u_i \otimes u_i \otimes u_i \): \( u_i \perp u_j \) for \( i \neq j \).

Decomposition through Tensor Norm Max.

\[
\max_v T(v, v, v), \|v\| = 1, v \in \mathbb{R}^d.
\]

- Lagrangian: \( T(v, v, v) - 1.5\lambda(\|v\|^2 - 1) \).
- Critical points: \( T(v, v, \cdot) = \lambda v \).
- No. of eigenvectors: \( \exp(d) \)!
- All saddle points are non-degenerate.
- Local optima: \( \{u_i\} \) for \( i = 1, \ldots, k \).

Multiple local optima, but they correspond to components!

Exponentially many saddle points!
Symmetric Tensor Decomposition

\[ T = v_1 \otimes^3 + v_2 \otimes^3 + \cdots, \]

Symmetric Tensor Decomposition

Tensor Power Method

\[ v \mapsto \frac{T(v, v, \cdot)}{\|T(v, v, \cdot)\|}. \]

\[ T(v, v, \cdot) = \langle v, v_1 \rangle^2 v_1 + \langle v, v_2 \rangle^2 v_2 \]

Symmetric Tensor Decomposition

Tensor Power Method

\[ v \mapsto \frac{T(v, v, \cdot)}{\|T(v, v, \cdot)\|}. \]

\[ T(v, v, \cdot) = \langle v, v_1 \rangle^2 v_1 + \langle v, v_2 \rangle^2 v_2 \]

Orthogonal Tensors

- \( \vec{v}_1 \perp \vec{v}_2 \).
- \( T(v_1, v_1, \cdot) = \lambda_1 v_1 \).

Symmetric Tensor Decomposition

Tensor Power Method

\[ v \mapsto \frac{T(v, v, \cdot)}{\|T(v, v, \cdot)\|} \]

\[ T(v, v, \cdot) = \langle v, v_1 \rangle^2 v_1 + \langle v, v_2 \rangle^2 v_2 \]

Orthogonal Tensors

- \( \vec{v}_1 \perp \vec{v}_2 \).
- \( T(v_1, v_1, \cdot) = \lambda_1 v_1 \).

Symmetric Tensor Decomposition

Tensor Power Method

\[ v \mapsto \frac{T(v, v, \cdot)}{\|T(v, v, \cdot)\|}. \]

\[ T(v, v, \cdot) = \langle v, v_1 \rangle^2 v_1 + \langle v, v_2 \rangle^2 v_2 + \cdots \]

Exponential no. of stationary points for power method:
\[ T(v, v, \cdot) = \lambda v \]

Symmetric Tensor Decomposition

Tensor Power Method

\[ v \mapsto \frac{T(v, v, \cdot)}{\|T(v, v, \cdot)\|} \]

\[ T(v, v, \cdot) = \langle v, v_1 \rangle^2 v_1 + \langle v, v_2 \rangle^2 v_2 \]

Exponential no. of stationary points for power method:

\[ T(v, v, \cdot) = \lambda v \]

Stable

Unstable

Other stationary points

Symmetric Tensor Decomposition

Tensor Power Method

\[ v \mapsto \frac{T(v, v, \cdot)}{\|T(v, v, \cdot)\|}. \]

\[ T(v, v, \cdot) = \langle v, v_1 \rangle^2 v_1 + \langle v, v_2 \rangle^2 v_2 \]

Exponential no. of stationary points for power method:
\[ T(v, v, \cdot) = \lambda v \]

For power method on orthogonal tensor, no spurious stable points

Implication: Guaranteed Tensor Decomposition

Given Orthogonal Tensor $T = \sum_{i \in [k]} \lambda_i u_i \otimes u_i \otimes u_i$.

Recover components one by one

- Run projected SGD on $\max_{v: \|v\| = 1} T(v, v, v)$.
- Guaranteed to recover a local optimum $\{u_i\}$ (upto scale).
- Find all components $\{u_i\}$ by deflation!
Implication: Guaranteed Tensor Decomposition

Given Orthogonal Tensor $T = \sum_{i \in [k]} \lambda_i u_i \otimes u_i \otimes u_i$.

Recover components one by one

- Run projected SGD on $\max_{v: \|v\| = 1} T(v, v, v)$.
- Guaranteed to recover a local optimum $\{u_i\}$ (upto scale).
- Find all components $\{u_i\}$ by deflation!

Alternative: simultaneous recovery of components (Ge et al ‘15)

- For fourth order tensor $T$ $\min_{\forall i, \|v_i\| = 1} \sum_{i \neq j} T(v_i, v_i, v_j, v_j)$.
- All saddle points are non-degenerate.
- All local optima are global.

SGD recovers the orthogonal tensor components optimally
Perturbation Analysis for Tensor Decomposition

- Well understood for matrix decomposition vs. hard for polynomials.
- Subtle analysis for tensor decomposition.

Perturbation Analysis for Tensor Decomposition

- Well understood for matrix decomposition vs. hard for polynomials.
- Subtle analysis for tensor decomposition.

\[ T \in \mathbb{R}^{d \times d \times d}: \text{Orthogonal tensor. } E: \text{noise tensor.} \]

\[ \hat{T} = T + E, \quad T = \sum_i \lambda_i v_i \otimes v_i \otimes v_i, \quad \|E\| := \max_{x: \|x\|=1} |E(x, x, x)|. \]

**Theorem:** When \( \|E\| < \frac{\lambda_{\min}}{\sqrt{d}} \), power method recovers \( \{v_i\} \) up to error \( \|E\| \) with linear no. of restarts.

Theorem: When $\|E\| < \frac{\lambda_{\min}}{\sqrt{d}}$, power method recovers $\{v_i\}$ up to error $\|E\|$ with linear no. of restarts.

Non-orthogonal Tensor Decomposition

\[ T = v_1 \otimes^3 + v_2 \otimes^3 + \cdots \]

Non-orthogonal Tensor Decomposition

Orthogonalization

Input tensor $T$

Non-orthogonal Tensor Decomposition

Orthogonalization

\[ T(W, W, W) = \tilde{T} \]

Non-orthogonal Tensor Decomposition

Orthogonalization

\[ \tilde{v}_1 \tilde{v}_2 = T(W, W, W) = \tilde{T} \]

Non-orthogonal Tensor Decomposition

Orthogonalization

\[ T(W, W, W) = \tilde{T} \]

\[ \tilde{T} = T(W, W, W) = \tilde{v}_1 \otimes^3 + \tilde{v}_2 \otimes^3 + \cdots \]

Non-orthogonal Tensor Decomposition

Orthogonalization

\[ v_1 \ v_2 \ W \ = \ \tilde{v}_1 \ \tilde{v}_2 \]

\[ T(W, W, W) = \tilde{T} \]

Find \( W \) using SVD of Matrix Slice

\[ M = T(\cdot, \cdot, \theta) = \]

Orthogonalization: invertible when $v_i$'s linearly independent.

Recap: Basic Tensor Decomposition Method

Toy Example in MATLAB

- Simulated Samples: Exchangeable Model
- Whiten The Samples
  - Second Order Moments
  - Matrix Decomposition
- Orthogonal Tensor Eigen Decomposition
  - Third Order Moments
  - Power Iteration
Simulated Samples: Exchangeable Model

Model Parameters

- **Hidden State:**
  \[ h \in \text{basis} \{ e_1, \ldots, e_k \} \]
  \[ k = 2 \]

- **Observed States:**
  \[ x_i \in \text{basis} \{ e_1, \ldots, e_d \} \]
  \[ d = 3 \]

- **Conditional Independency:**
  \[ x_1 \perp \perp x_2 \perp \perp x_3 | h \]

- **Transition Matrix:** \( A \)

- **Exchangeability:**
  \[ \mathbb{E}[x_i | h] = Ah, \forall i \in 1, 2, 3 \]
Simulated Samples: Exchangeable Model

Model Parameters

- **Hidden State:**
  \[ h \in \text{basis}\{e_1, \ldots, e_k\} \]
  \[ k = 2 \]

- **Observed States:**
  \[ x_i \in \text{basis}\{e_1, \ldots, e_d\} \]
  \[ d = 3 \]

- **Conditional Independence:**
  \[ x_1 \perp x_2 \perp x_3 | h \]

- **Transition Matrix:** \( A \)

- **Exchangeability:**
  \[ \mathbb{E}[x_i|h] = Ah, \forall i \in 1,2,3 \]

---

**Generate Samples Snippet**

```matlab
for t = 1 : n
    % generate h for this sample
    h_category=(rand()>0.5) + 1;
    h(t,h_category)=1;
    transition_cum=cumsum(A_true(:,h_category));
    % generate x1 for this sample | h
    x_category=find(transition_cum>rand(),1);
    x1(t,x_category)=1;
    % generate x2 for this sample | h
    x_category=find(transition_cum >rand(),1);
    x2(t,x_category)=1;
    % generate x3 for this sample | h
    x_category=find(transition_cum >rand(),1);
    x3(t,x_category)=1;
end
```
Whiten The Samples

Second Order Moments
- \( M_2 = \frac{1}{n} \sum_t x_1^t \otimes x_2^t \)

Whitening Matrix
- \( W = U_w L_w^{-0.5}, \)
  \[ [U_w, L_w] = \text{k-svd}(M_2) \]

Whiten Data
- \( y_1^t = W^\top x_1^t \)

Orthogonal Basis
- \( V = W^\top A \rightarrow V^\top V = I \)
Orthogonal Tensor Eigen Decomposition

Third Order Moments

\[ T = \frac{1}{n} \sum_{t \in [n]} y_1^t \otimes y_2^t \otimes y_3^t \approx \sum_{i \in [k]} \lambda_i v_i \otimes v_i \otimes v_i, \quad V^TV = I \]

Gradient Ascent

\[ T(I, v_1, v_1) = \frac{1}{n} \sum_{t \in [n]} \langle v_1, y_2^t \rangle \langle v_1, y_3^t \rangle y_1^t \approx \sum_i \lambda_i \langle v_i, v_1 \rangle^2 v_i = \lambda_1 v_1. \]

\( v_i \) are eigenvectors of tensor \( T \).
Orthogonal Tensor Eigen Decomposition

\[ T \leftarrow T - \sum_j \lambda_j v_j^{\otimes 3}, \quad v \leftarrow \frac{T(I, v, v)}{\|T(I, v, v)\|} \]

Power Iteration Snippet

```matlab
V = zeros(k,k); Lambda = zeros(k,1);
for i = 1:k
    v_old = rand(k,1); v_old = normc(v_old);
    for iter = 1 : Maxiter
        v_new = (y1' * ((y2*v_old).*(y3*v_old)))/n;
        if i > 1
            % deflation
            for j = 1: i-1
                v_new = v_new - (V(:,j)*(v_old'*V(:,j))^2)* Lambda(j);
            end
        end
        lambda = norm(v_new); v_new = normc(v_new);
        if norm(v_old - v_new) < TOL
            fprintf('Converged at iteration %d.
', iter);
            V(:,i) = v_new; Lambda(i,1) = lambda;
            break;
        end
        v_old = v_new;
    end
end
```
Orthogonal Tensor Eigen Decomposition

\[ T \leftarrow T - \sum_j \lambda_j v_j \otimes^3, \quad v \leftarrow \frac{T(I, v, v)}{\|T(I, v, v)\|} \]

Power Iteration Snippet

```matlab
V = zeros(k,k); Lambda = zeros(k,1);
for i = 1:k
    v_old = rand(k,1); v_old = normc(v_old);
    for iter = 1 : Maxiter
        v_new = (y1*(y2*v_old).*y3*v_old))/n;
        if i > 1
            v_new = v_new - (V(:,j)*(v_old'*V(:,j))^2)* Lambda(j);
        end
        lambda = norm(v_new); v_new = normc(v_new);
        if norm(v_old - v_new) < TOL
            fprintf('Converged at iteration %d.', iter);
            V(:,i) = v_new; Lambda(i,1) = lambda;
            break;
        end
        v_old = v_new;
    end
end
```

Green: Groundtruth
Red: Estimation at each iteration
Automated Categorization of Documents

Mixed topics

At Florida State, Football Clouds Justice

Now, an examination by The New York Times of police and court records, along with interviews with crime witnesses, has found that, far from an aberration, the treatment of the Winston complaint was in keeping with the way the police on numerous occasions have soft-pedaled allegations of wrongdoing by Seminoles football players. From criminal mischief and motor-vehicle theft to domestic violence, arrests have been avoided, investigations have stalled and players have escaped serious consequences.

In a community whose self-image and economic well-being are so tightly bound to the fortunes of the nation’s top-ranked college football team, law enforcement officers are finely attuned to a suspect’s football connections. Those ties are cited repeatedly in police reports examined by The Times. What’s more, dozens of officers work second jobs directing traffic and providing security at home football games, and many express their devotion to the Seminoles on social media.

On Jan. 10, 2013, a female student at Florida State spotted the man she believed had raped her the previous month. After learning his name, Jameis Winston, she reported him to the Tallahassee police.

In the 21 months since, Florida State officials have said little about how they handled the case, which is no investigated by the federal Department of Justice. It did not become public until November, when a Tampa reporter, Matt Baker, acting on a tip, sought records of the police investigation.

Most recently, university officials suspended Mr. Winston for one game after he stood in a public place on campus and played off a running Internet gag, shouting a crude reference to a sex act. In a news conference afterward, his coach, Jimbo Fisher, said, “Our hope and belief is Jameis will learn from this and use better judgment and language and decision-making.”

Upon learning of Mr. Baker’s inquiry, Florida State, having shown little curiosity about the rape accusation, suddenly took a keen interest in the journalist seeking to report it, according to emails obtained by The Times.

“Can you share any details on the requesting source?” David Perry, the university’s police chief, asked the Tallahassee police. Several hours later, Mr.
Tensor Methods for Topic Modeling

- Topic-word matrix $\mathbb{P}[\text{word} = i | \text{topic} = j]$
- Linearly independent columns

Moment Tensor: Co-occurrence of Word Triplets
Visualization

http://newport.eecs.uci.edu/anandkumar/Lab/Lab_sub/NewYorkTimes
Tensor Methods for Community Modeling

Mixed Membership Stochastic Block Model

- Edges \textit{conditionally independent} given community memberships.
- Community membership proportions $\sim \text{Dir}(\alpha)$.

Extracting Communities in Social Networks

Moment Tensor: Common Friends among Node Triplets

Tensors vs. Variational Inference

Criterion: Perplexity = \exp[-\text{likelihood}].

Learning Topics from PubMed on Spark, 8mil articles

Tensors vs. Variational Inference
Criterion: Perplexity = \( \exp[-\text{likelihood}] \).

Learning Topics from PubMed on Spark, 8mil articles

Learning network communities from social network data
Facebook \( n \sim 20k \), Yelp \( n \sim 40k \), DBLP-sub \( n \sim 1e5 \), DBLP \( n \sim 1e6 \).
Tensors vs. Variational Inference

Criterion: Perplexity $= \exp[-\text{likelihood}].$

Learning Topics from PubMed on Spark, 8mil articles

Learning network communities from social network data

Facebook $n \sim 20k$, Yelp $n \sim 40k$, DBLP-sub $n \sim 1e5$, DBLP $n \sim 1e6$.

Orders of Magnitude Faster & More Accurate

Sample Screenshot

- Infrastructure: AWS 6 nodes with 36 cores each.
- Data: PubMed 8.1 million articles, 140k vocabulary. 50 topics.
- Total runtime: 27 mins.
- https://github.com/FurongHuang/SpectralLDA-TensorSpark
Goal

- Cataloging the **neuronal cell types**
- Inferring **gene expression profiles** of cell types
Cataloging Neuronal Cell Types In the Brain

Goal
- Cataloging the neuronal cell types
- Inferring gene expression profiles of cell types

Data
- Allen Brain Atlas: Mouse Brain
- Cellular resolution in-situ hybridization image
- Mis-alignment

Big data: 64 TB scale data
Feature Extraction - From Image to Cell

- Previous: Average Expression Level / Volume
- Ours: Measure distribution of neuron features
  - size, orientation, counts in a unit radius, gene expression level
Feature Extraction - From Image to Cell

- Previous: Average Expression Level / Volume
- Ours: Measure distribution of neuron features
  - size, orientation, counts in a unit radius, gene expression level

Detect Cells

(a) Patch from gene Pvalb slice

(b) Cell detection and extraction
Discover Cell Types via **Un-mixing** of Spatial Point Process Mixtures

- Image/Gene is a mixture of cell types
- Cell is a mixture of genes
# Discoveries via Unsupervised Learning

<table>
<thead>
<tr>
<th>Type</th>
<th>Top Genes</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Cell-type 1</td>
<td>Gfap</td>
<td>Itga3</td>
<td>Asb4</td>
<td>Actr2</td>
</tr>
<tr>
<td>Cell-type 2</td>
<td>Sst</td>
<td>Gad2</td>
<td>Gad1</td>
<td>Ifnar1</td>
</tr>
<tr>
<td>Cell-type 3</td>
<td>Bhlhe22</td>
<td>Ptgds</td>
<td>Srgap3</td>
<td>Cry2</td>
</tr>
</tbody>
</table>

1. astrocytes
2. interneurons
3. oligodendrocytes
Discoveries via Unsupervised Learning

<table>
<thead>
<tr>
<th>Type</th>
<th>Top Genes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cell-type 1</td>
<td>Gfap, Itga3, Asb4, Actr2</td>
</tr>
<tr>
<td>Cell-type 2</td>
<td>Sst, Gad2, Gad1, Ifnar1</td>
</tr>
<tr>
<td>Cell-type 3</td>
<td>Bhlhe22, Ptgds, Srgap3, Cry2</td>
</tr>
</tbody>
</table>

1. astrocytes
2. interneurons
3. oligodendrocytes

**Discovered cell feature distribution**

**Size**

Cell Fraction

**Gene expression level**

Cell Fraction

**Orientation**

Cells in unit radius

Hierarchical Tensor Decomposition

\[ \begin{align*}
&= F + F + F
\end{align*} \]
Hierarchical Tensor Decomposition

\[ = \mathbf{F} + \mathbf{F} + \mathbf{F} + \mathbf{F} + \mathbf{F} + \mathbf{F} + \mathbf{F} + \mathbf{F} \]
Hierarchical Tensor Decomposition
Hierarchical Tensor Decomposition

Latent tree graphical model: less restrictive

- Versatile in modeling hierarchical relations
- Number and location of hidden variables unknown
Latent Tree Model Learning - Structure

Structure & Parameter Learning: Divide and Conquer

- “Divide-and-conquer” strategy
“Divide-and-conquer” strategy

- Find local groups: MST
Latent Tree Model Learning - Structure

Structure & Parameter Learning: Divide and Conquer

- “Divide-and-conquer” strategy
  - Find local groups: MST
  - Learn over small groups
“Divide-and-conquer” strategy

- Find local groups: MST
- Learn over small groups
“Divide-and-conquer” strategy
- Find local groups: MST
- Learn over small groups
Latent Tree Model Learning - Structure

Structure & Parameter Learning: Divide and Conquer

“Divide-and-conquer” strategy
- Find local groups: MST
- Learn over small groups
“Divide-and-conquer” strategy
- Find local groups: MST
- Learn over small groups
- Merge into global solution
"Divide-and-conquer" strategy

- Find local groups: MST
- Learn over small groups
- Merge into global solution
Latent Tree Model Learning - Structure

Structure & Parameter Learning: Divide and Conquer

- “Divide-and-conquer” strategy
  - Find local groups: MST
  - Learn over small groups
  - Merge into global solution

- Local method w. global consistency guarantee
“Divide-and-conquer” strategy
- Find local groups: MST
- Learn over small groups
- Merge into global solution

Local method w. global consistency guarantee

Bulk asynchronous parallel
- $\log(\#\text{ variables});\text{ linear}(\text{dimension})$
Latent Tree for Discovering Human Disease Hierarchy

CMS: 1.6 million patients, 168 million diagnostic events, 11 k diseases.

Overcomplete Dictionary learning

Sparse coding prevalent in neural signaling.

Neural sparse coding [Papadopoulou11]

Overcomplete Dictionary learning

Sparse coding prevalent in neural signaling.

Neural sparse coding [Papadopoulou11]


Overcomplete Dictionary learning

Contribution: learn overcomplete incoherent dictionaries

<table>
<thead>
<tr>
<th>Neural sparse coding [Papadopoulou11]</th>
<th>Linear Model with Overcomplete Dictionary</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Neural sparse coding" /></td>
<td><img src="image" alt="Linear Model with Overcomplete Dictionary" /></td>
</tr>
</tbody>
</table>


Overcomplete Dictionary learning

Contribution: learn overcomplete incoherent dictionaries

- Incoherent dictionary: rules out redundancy.
- Blessing of high dimension: exponential no. of vectors!

Neural sparse coding [Papadopoulou11]  

Linear Model with Overcomplete Dictionary


Overcomplete Dictionary learning

Contribution: learn overcomplete incoherent dictionaries

- Power method for overcomplete tensor decomposition.
- Initialization algorithms for alternating minimization.

Neural sparse coding
[Neural sparse coding source]

Linear Model with Overcomplete Dictionary


Convolutional Dictionary learning


Convolutional Dictionary learning

Shift-invariant Dictionary

Image

Dictionary


Convolutional Dictionary learning

Shift-invariant Dictionary

Image

Dictionary

Convolulotional Model

= +

Convolutional Dictionary learning

Shift-invariant Dictionary

Image

Dictionary

Linear Model with Constraints


Convolutional Dictionary learning

Efficient Tensor Decomposition with Shifted Components

\[ \text{Shift-invariant Dictionary} \]

\[ \text{Linear Model with Constraints} \]


Learning Representations

Sparse coding prevalent in neural signaling.

Neural sparse coding
[Papadopoulou11]


Learning Representations

Sparse coding prevalent in neural signaling.

Neural sparse coding [Papadopoulou11]

Linear Model with Overcomplete Dictionary

Learning Representations

Contribution: learn overcomplete incoherent dictionaries

Neural sparse coding [Papadopoulou11]

Linear Model with Overcomplete Dictionary


Moment forms for Dictionary Models
\[ x_i = Ah_i, \quad i \in [n]. \]

Independent components analysis (ICA)

- \( h_i \) are independent, e.g. Bernoulli Gaussian

\[
M_4 := \mathbb{E}[x \otimes x \otimes x \otimes x] - T,
\]

where

\[
T_{i_1,i_2,i_3,i_4} := \mathbb{E}[x_{i_1} x_{i_2}] \mathbb{E}[x_{i_3} x_{i_4}] + \mathbb{E}[x_{i_1} x_{i_3}] \mathbb{E}[x_{i_2} x_{i_4}] + \mathbb{E}[x_{i_1} x_{i_4}] \mathbb{E}[x_{i_2} x_{i_3}],
\]

Let \( \kappa_j := \mathbb{E}[h_j^4] - 3\mathbb{E}^2[h_j^2], \ j \in [k]. \) Then, we have

\[
M_4 = \sum_{j \in [k]} \kappa_j a_j \otimes a_j \otimes a_j \otimes a_j.
\]
Moment forms for Dictionary Models

General (sparse) coefficients

\[ x_i = A h_i, \quad i \in [n], \quad \mathbb{E}[h_i] = s. \]

\[ \mathbb{E}[h_i^4] = \mathbb{E}[h_i^2] = \beta s/k, \]

\[ \mathbb{E}[h_i^2 h_j^2] \leq \tau, \quad i \neq j, \]

\[ \mathbb{E}[h_i^3 h_j] = 0, \quad i \neq j, \]

\[ \mathbb{E}[x \otimes x \otimes x \otimes x] = \sum_{j \in [k]} \kappa_j a_j \otimes a_j \otimes a_j \otimes a_j + E, \text{ where } \|E\| \leq \tau \|A\|^4. \]

\[ = \begin{array}{c}
\text{red}
\end{array} + \begin{array}{c}
\text{red}
\end{array} \cdots \]
Convolutional Dictionary Model

- So far, invariances in dictionary are not incorporated.
- Convolutional models: incorporate invariances such as shift invariance.

Image

Dictionary elements
Rewriting as a standard dictionary model

(a) Convolutional model

\[ x = \sum_i f_i * w_i = \sum_i \text{Cir}(f_i)w_i = \mathcal{F}^* w^* \]

- Circulant matrix has eigen decomposition
  \[ \text{Cir}(f) = U \text{Diag}(\text{DFT}_{1-d}(f)) U^H = U \text{Diag}(\sqrt{n} U^H \cdot f) U^H \]
  - \( U \) is the Discrete Fourier Transform Matrix.

(b) Reformulated model
Moment forms and optimization

\[ x = \sum_i f_i \ast w_i = \sum_i \text{Cir}(f_i)w_i = \mathcal{F}^*w^* \]

- Assume coefficients \( w_i \) are independent (convolutional ICA model)
- Cumulant tensor has decomposition with components \( \mathcal{F}_i^* \).
Analysis

\[ x = f_1^* w_1^* + f_L^* w_L^* \]

Comparison with Alternating Minimization (AM) method:

<table>
<thead>
<tr>
<th>Methods</th>
<th>Running Time</th>
<th>Processors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tensor Factorization</td>
<td>(O(\log(n) + \log(L)))</td>
<td>(O(L^2 n^3))</td>
</tr>
<tr>
<td>AM</td>
<td>(O(\max(\log(n)\log(L), \log(n)\log(N))))</td>
<td>(O(\max(\frac{nNL}{\log N}, \frac{nNL}{\log L})))</td>
</tr>
</tbody>
</table>

Table: Computation complexity (\(L\) is the number of filters, \(n\) is the dimension of filters. \(N\) is the number of samples)
Analysis

- Non-convex optimization: guaranteed convergence to local optimum
- Local optima are shifted filters
Application: Paraphrase Detection

Microsoft paraphrase data: 5800 pairs of sentences

One-hot encoding matrix
Application: Paraphrase Detection

Microsoft paraphrase data: 5800 pairs of sentences

- PCA on One-hot Encoding Matrix $\rightarrow$ Subspace and Projected data
Application: Paraphrase Detection
Microsoft paraphrase data: 5800 pairs of sentences

- Principal components
- One-hot encoding matrix
- Projected data

CT on each coordinate $\rightarrow$ activation map for each coordinate
Application: Paraphrase Detection
Microsoft paraphrase data: 5800 pairs of sentences

Stack all activation maps → Sentence Embedding
Application: Paraphrase Detection

Microsoft paraphrase data: 5800 pairs of sentences

- Detects from scratch (unsupervised).
- Incorporates context.
Results using Sentence Embeddings

Sentiment Analysis

<table>
<thead>
<tr>
<th>Method</th>
<th>MR</th>
<th>SUBJ</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNB</td>
<td>79.0</td>
<td>93.6</td>
</tr>
<tr>
<td>Paragraph-vector</td>
<td>74.8</td>
<td>90.5</td>
</tr>
<tr>
<td>Skip-thought</td>
<td>75.5</td>
<td>92.1</td>
</tr>
<tr>
<td>ConvDic+DeconvDec</td>
<td>78.9</td>
<td>92.4</td>
</tr>
</tbody>
</table>

Paraphrase Detection

<table>
<thead>
<tr>
<th>Method</th>
<th>Outside Information</th>
<th>F score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vector Similarity</td>
<td>word similarity</td>
<td>75.3%</td>
</tr>
<tr>
<td>RMLMG</td>
<td>syntacticinfo</td>
<td>80.5%</td>
</tr>
<tr>
<td>ConvDic+DeconvDec</td>
<td>none</td>
<td>80.7%</td>
</tr>
<tr>
<td>Skip-thought</td>
<td>book corpus</td>
<td>81.9%</td>
</tr>
</tbody>
</table>
Image Pattern Learning through Tensor Factorization

- 2-D Convolutional Model

\[ X = \sum_{j=1}^{L} V_j^* \ast W_j^* \]

Slides in this section prepared by Y. Shi
Image Pattern Learning through Tensor Factorization

Key points:

- Recall: 1-D circulant matrix eigen decomposition corresponds to 1-D Discrete Fourier Transform
Key points:

- Recall: 1-D circulant matrix eigen decomposition corresponds to 1-D Discrete Fourier Transform.

- 2-D circulant matrix eigen decomposition:
  \[ \text{Cir}_{2-d}(V) = (U \otimes U)\text{Diag}(\text{DFT}_{2-d}(V))(U \otimes U)^H \]

- The eigenvector matrix for 2-D circulant matrix is \( U \otimes U \).
Main algorithm:

1. Subsample and batch
2. Form third order cumulant tensor
3. Tensor factorization
**MPEG7 Dataset**

- Image/activation map size: 28 X 28, filter size: 10 X 10
- First layer filters.

<table>
<thead>
<tr>
<th>Images</th>
<th>Filters</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Filter 1" /></td>
<td><img src="image1_1.png" alt="Filter 1 Image 1" /></td>
</tr>
<tr>
<td><img src="image2.png" alt="Filter 2" /></td>
<td><img src="image2_1.png" alt="Filter 2 Image 1" /></td>
</tr>
<tr>
<td><img src="image3.png" alt="Filter 3" /></td>
<td><img src="image3_1.png" alt="Filter 3 Image 1" /></td>
</tr>
<tr>
<td><img src="image4.png" alt="Filter 4" /></td>
<td><img src="image4_1.png" alt="Filter 4 Image 1" /></td>
</tr>
</tbody>
</table>
**MPEG7 Dataset**

- Image/activation map size: 28 X 28, filter size: 10 X 10
- Second layer filters after max-pooling.

<table>
<thead>
<tr>
<th>Images</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Image 1" /></td>
</tr>
<tr>
<td><img src="image2.png" alt="Image 2" /></td>
</tr>
<tr>
<td><img src="image3.png" alt="Image 3" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Filters</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="filter1.png" alt="Filter 1" /></td>
</tr>
<tr>
<td><img src="filter2.png" alt="Filter 2" /></td>
</tr>
<tr>
<td><img src="filter3.png" alt="Filter 3" /></td>
</tr>
<tr>
<td><img src="filter4.png" alt="Filter 4" /></td>
</tr>
<tr>
<td><img src="filter5.png" alt="Filter 5" /></td>
</tr>
</tbody>
</table>

---

**Images**

- ![Image 1](image1.png)
- ![Image 2](image2.png)
- ![Image 3](image3.png)

**Filters**

- ![Filter 1](filter1.png)
- ![Filter 2](filter2.png)
- ![Filter 3](filter3.png)
- ![Filter 4](filter4.png)
- ![Filter 5](filter5.png)
Training Neural Networks with Tensors

Training Neural Networks with Tensors

Given input pdf $p(\cdot)$, 

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

Gaussian $x \Rightarrow$ Hermite polynomials.

Rewards from hidden state.
- Actions drive hidden state evolution.

**Reinforcement Learning of POMDPs**

Learning in Adaptive Environments
Reinforcement Learning of POMDPs

Learning in Adaptive Environments

- Rewards from hidden state.
- Actions drive hidden state evolution.

Partially Observable Markov Decision Process

Learning using tensor methods under memoryless policies
POMDP model with 3 hidden states (trained using tensor methods) vs. NN with 3 hidden layers 10 neurons each (trained using RmsProp).
POMDP model with 8 hidden states (trained using tensor methods) vs. NN with 3 hidden layers 30 neurons each (trained using RmsProp).
POMDP model with 8 hidden states (trained using tensor methods) vs. NN with 3 hidden layers 30 neurons each (trained using RmsProp). Faster convergence to better solution via tensor methods.
Outline

1. Introduction
2. Why Tensors?
3. Algorithms for Tensor Decomposition
4. Learning Probabilistic Models
5. Learning Representations
6. Conclusion
Conclusion

Guaranteed Unsupervised Learning

- Matrix and tensor methods have desirable guarantees on reaching global optimum.
- Method of moment estimators using tensor decomposition.
- Polynomial computational and sample complexity.
- Faster and better performance in practice.

Steps Forward

- Scaling up tensor methods: sketching algorithms, extended BLAS, ...
- Unified conditions on when unsupervised learning is tractable?
Resources for Today

- Tensor introduction
  http://newport.eecs.uci.edu/anandkumar/tensor.html

- Fast tensor decomposition algorithms.
  https://bitbucket.org/megaDataLab/tensormethodsforml/overview

- Spectral methods for training latent Dirichlet allocation topic models on Apache Spark.
  https://github.com/FurongHuang/SpectralLDA-TensorSpark