Beating the Perils of Non-Convexity: Machine Learning using Tensor Methods

Anima Anandkumar


Joint work with Majid Janzamin and Hanie Sedghi.

U.C. Irvine

Learning with Big Data

Learning is finding needle in a haystack

Learning with Big Data

Learning is finding needle in a haystack

- High dimensional regime: as data grows, more variables!
- Useful information: low-dimensional structures.
- Learning with big data: ill-posed problem.

Learning with Big Data

Learning is finding needle in a haystack

- High dimensional regime: as data grows, more variables!
- Useful information: low-dimensional structures.
- Learning with big data: ill-posed problem.

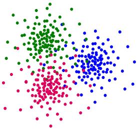
- Learning with big data: statistically and computationally challenging!

Optimization for Learning

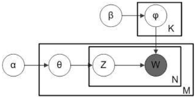
Most learning problems can be cast as optimization.

Unsupervised Learning

- Clustering
  k-means, hierarchical . . .

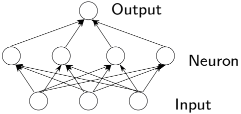


- Maximum Likelihood Estimator
  Probabilistic latent variable models



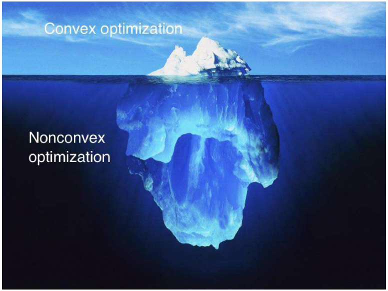
Supervised Learning

- Optimizing a neural network with respect to a loss function



Convex vs. Non-convex Optimization

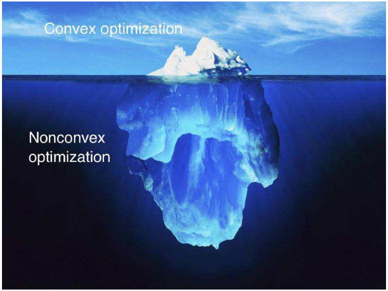
Progress is only tip of the iceberg..



Images taken from `https://www.facebook.com/nonconvex`

Convex vs. Non-convex Optimization

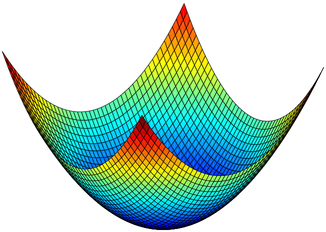
Progress is only tip of the iceberg..

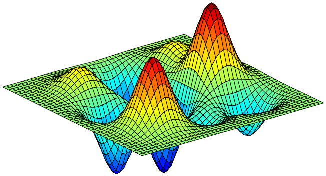


Real world is mostly non-convex!

Images taken from https://www.facebook.com/nonconvex

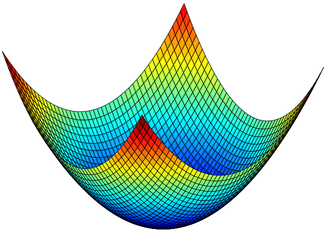
Convex vs. Nonconvex Optimization



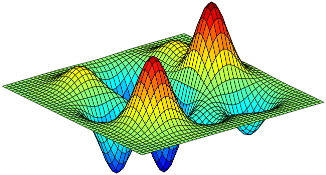


- Unique optimum: global/local.
- Multiple local optima

Convex vs. Nonconvex Optimization

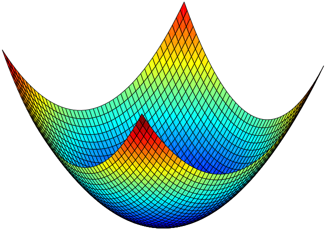


- Unique optimum: global/local.

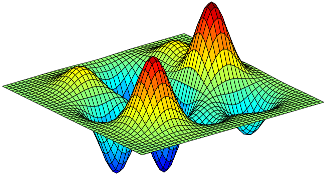


- Multiple local optima
- In high dimensions possibly exponential local optima

Convex vs. Nonconvex Optimization



- Unique optimum: global/local.



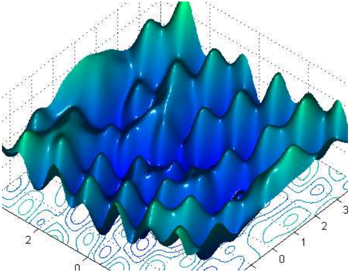
- Multiple local optima
- In high dimensions possibly exponential local optima

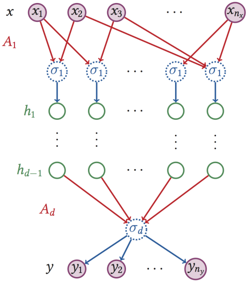
How to deal with non-convexity?

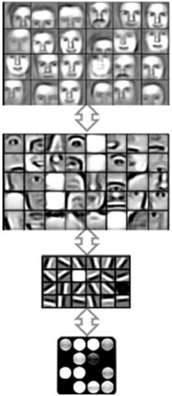
Outline

Training Neural Networks

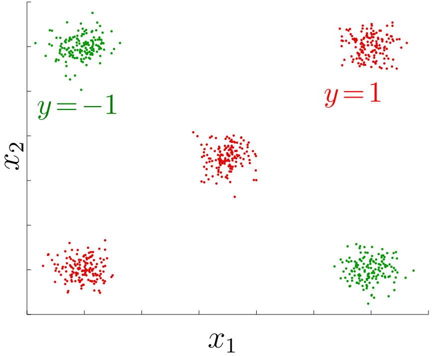
- Tremendous practical impact with deep learning.
- Algorithm: backpropagation.
- Highly non-convex optimization

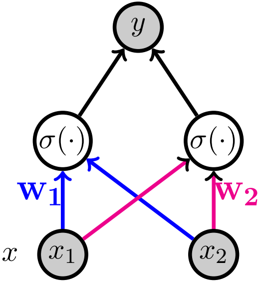






Toy Example: Failure of Backpropagation



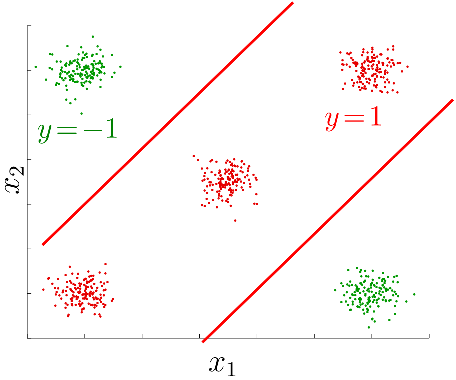


Labeled input samples

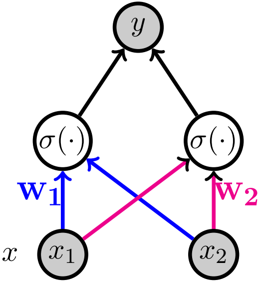
Goal: binary classification

Our method: guaranteed risk bounds for training neural networks

Toy Example: Failure of Backpropagation

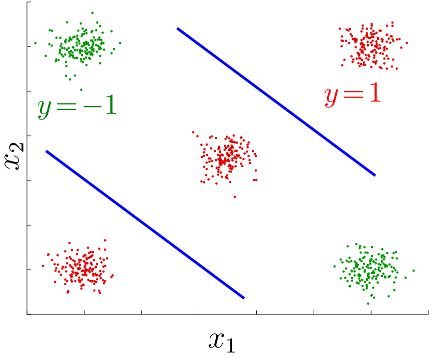


Labeled input samples
Goal: binary classification

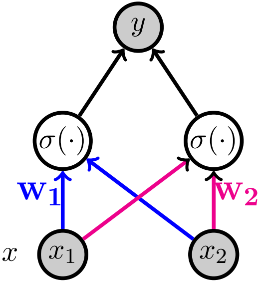


Our method: guaranteed risk bounds for training neural networks

Toy Example: Failure of Backpropagation



Labeled input samples
Goal: binary classification

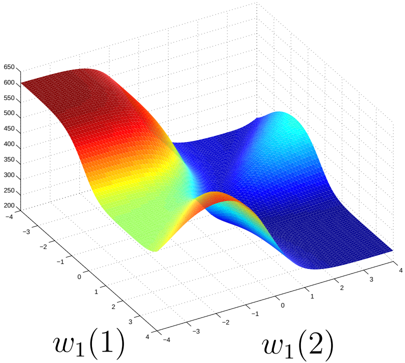


Our method: guaranteed risk bounds for training neural networks

Backpropagation vs. Our Method

Weights w_2 randomly drawn and fixed

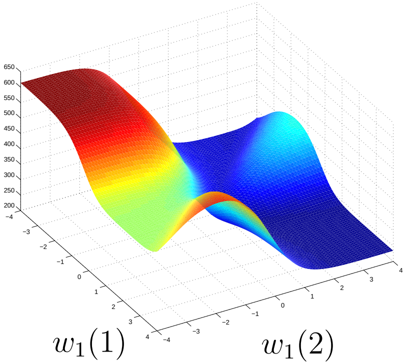
Backprop (quadratic) loss surface



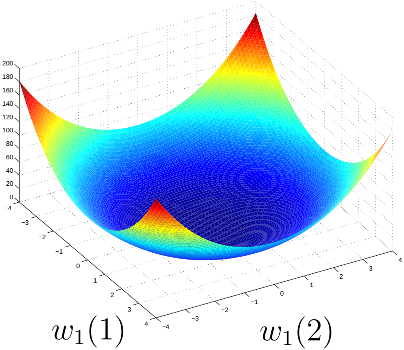
Backpropagation vs. Our Method

Weights w_2 randomly drawn and fixed

Backprop (quadratic) loss surface



Loss surface for our method



Overcoming Hardness of Training

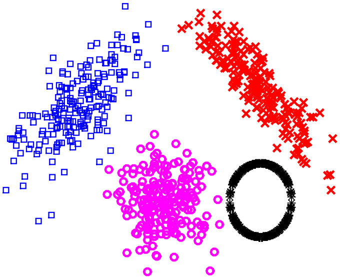
In general, training a neural network is NP hard.

How does knowledge of input distribution help?

Overcoming Hardness of Training

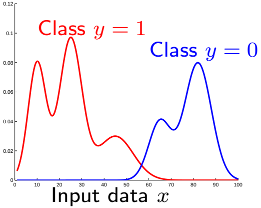
In general, training a neural network is NP hard.

How does knowledge of input distribution help?

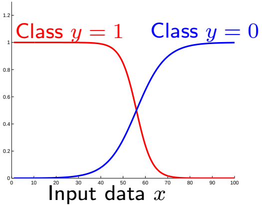


Generative vs. Discriminative Models

$p(x, y)$



$p(y|x)$



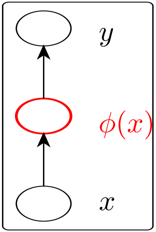
- Generative models: Encode domain knowledge.
- Discriminative: good classification performance.
- Neural Network is a discriminative model.

Do generative models help in discriminative tasks?

Feature Transformation for Training Neural Networks

Feature learning: Learn $\phi(\cdot)$ from input data.

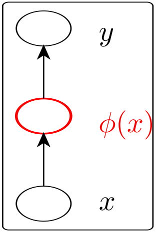
How to use $\phi(\cdot)$ to train neural networks?



Feature Transformation for Training Neural Networks

Feature learning: Learn $\phi(\cdot)$ from input data.

How to use $\phi(\cdot)$ to train neural networks?



Multivariate Moments: Many possibilities, ...

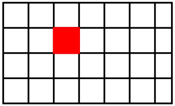
$$\mathbb{E}[x \otimes y], \quad \mathbb{E}[x \otimes x \otimes y], \quad \mathbb{E}[\phi(x) \otimes y], \quad \ldots$$

Tensor Notation for Higher Order Moments

- Multi-variate higher order moments form tensors.
- Are there spectral operations on tensors akin to PCA on matrices?

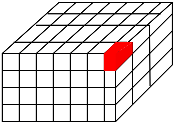
Matrix

- $\mathbb{E}[x \otimes y] \in \mathbb{R}^{d \times d}$ is a second order tensor.
- $\mathbb{E}[x \otimes y]_{i_1,i_2} = \mathbb{E}[x_{i_1} y_{i_2}]$.
- For matrices: $\mathbb{E}[x \otimes y] = \mathbb{E}[xy^\top]$.



Tensor

- $\mathbb{E}[x \otimes x \otimes y] \in \mathbb{R}^{d \times d \times d}$ is a third order tensor.
- $\mathbb{E}[x \otimes x \otimes y]_{i_1,i_2,i_3} = \mathbb{E}[x_{i_1} x_{i_2} y_{i_3}]$.

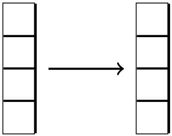


- In general, $\mathbb{E}[\phi(x) \otimes y]$ is a tensor.
- What class of $\phi(\cdot)$ useful for training neural networks?

Score Function Transformations

- Score function for $x \in \mathbb{R}^d$ with pdf $p(\cdot)$:

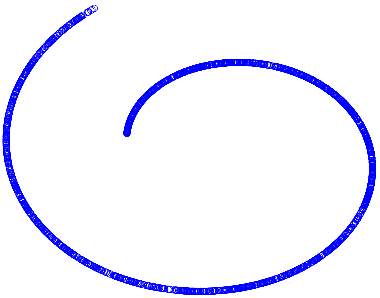
$$\boxed{\mathcal{S}_1(x) := -\nabla_x \log p(x)}$$

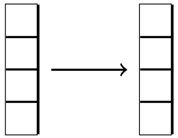


Score Function Transformations

- Score function for $x \in \mathbb{R}^d$ with pdf $p(\cdot)$:

$$\boxed{\mathcal{S}_1(x) := -\nabla_x \log p(x)}$$



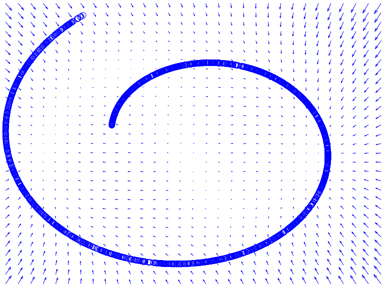


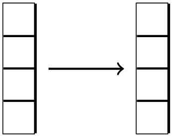
Input: $x \in \mathbb{R}^d$ $\quad$ $\mathcal{S}_1(x) \in \mathbb{R}^d$

Score Function Transformations

- Score function for $x \in \mathbb{R}^d$ with pdf $p(\cdot)$:

$$\boxed{\mathcal{S}_1(x) := -\nabla_x \log p(x)}$$





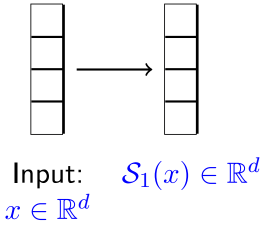
Input: $x \in \mathbb{R}^d$

$\mathcal{S}_1(x) \in \mathbb{R}^d$

Score Function Transformations

- Score function for $x \in \mathbb{R}^d$ with pdf $p(\cdot)$:

$$\boxed{\mathcal{S}_1(x) := -\nabla_x \log p(x)}$$

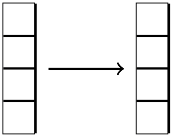


- m^{th}-order score function:

Score Function Transformations

- Score function for $x \in \mathbb{R}^d$ with pdf $p(\cdot)$:

$$\boxed{\mathcal{S}_1(x) := -\nabla_x \log p(x)}$$



Input: $x \in \mathbb{R}^d$

$\mathcal{S}_1(x) \in \mathbb{R}^d$

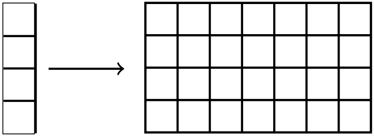
- m^{th}-order score function:

$$\boxed{\mathcal{S}_m(x) := (-1)^m \frac{\nabla^{(m)} p(x)}{p(x)}}$$

Score Function Transformations

- Score function for $x \in \mathbb{R}^d$ with pdf $p(\cdot)$:

$$\boxed{\mathcal{S}_1(x) := -\nabla_x \log p(x)}$$



- m^{th}-order score function:

$$\boxed{\mathcal{S}_m(x) := (-1)^m \frac{\nabla^{(m)} p(x)}{p(x)}}$$

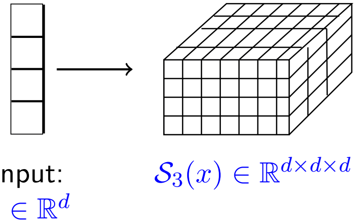
Score Function Transformations

- Score function for $x \in \mathbb{R}^d$ with pdf $p(\cdot)$:

$$\boxed{\mathcal{S}_1(x) := -\nabla_x \log p(x)}$$

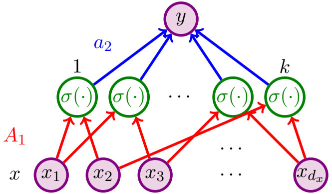
- m^{th}-order score function:

$$\boxed{\mathcal{S}_m(x) := (-1)^m \frac{\nabla^{(m)} p(x)}{p(x)}}$$

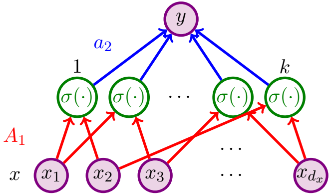


Moments of a Neural Network

$$\mathbb{E}[y|x] = f(x) = a_2^\top \sigma(A_1^\top x + b_1) + b_2$$



Moments of a Neural Network



$$\boxed{\mathbb{E}[y|x] = f(x) = a_2^\top \sigma(A_1^\top x + b_1) + b_2}$$

- Given labeled examples $\{(x_i, y_i)\}$

$$\mathbb{E}\left[y \cdot \mathcal{S}_m(x)\right] = \mathbb{E}\left[\nabla^{(m)} f(x)\right]$$

$$\Downarrow$$

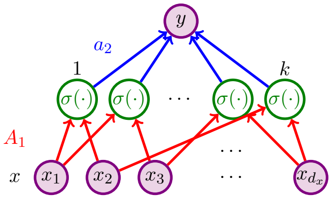
Moments of a Neural Network

$$\boxed{\mathbb{E}[y|x] = f(x) = a_2^\top \sigma(A_1^\top x + b_1) + b_2}$$

- Given labeled examples $\{(x_i, y_i)\}$

$$\mathbb{E}\left[y \cdot \mathcal{S}_m(x)\right] = \mathbb{E}\left[\nabla^{(m)} f(x)\right]$$

$$\Downarrow$$



$$\boxed{M_1 = \mathbb{E}[y \cdot \mathcal{S}_1(x)] = \sum_{j \in [k]} \lambda_{1,j} \cdot (A_1)_j}$$

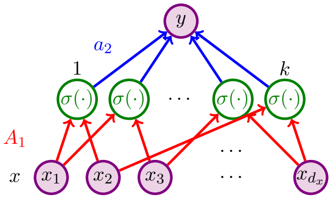
Moments of a Neural Network

$$\mathbb{E}[y|x] = f(x) = a_2^\top \sigma(A_1^\top x + b_1) + b_2$$

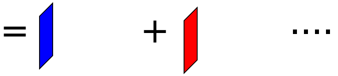
- Given labeled examples $\{(x_i, y_i)\}$

$$\mathbb{E}\left[y \cdot \mathcal{S}_m(x)\right] = \mathbb{E}\left[\nabla^{(m)} f(x)\right]$$

$$\Downarrow$$



$$M_1 = \mathbb{E}[y \cdot \mathcal{S}_1(x)] = \sum_{j \in [k]} \lambda_{1,j} \cdot (A_1)_j$$



$\lambda_{11}(A_1)_1$ $\qquad$ $\lambda_{12}(A_1)_2$

Moments of a Neural Network

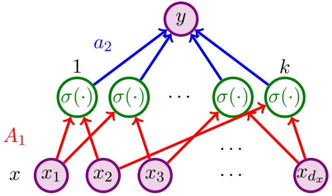
$$\boxed{\mathbb{E}[y|x] = f(x) = a_2^\top \sigma(A_1^\top x + b_1) + b_2}$$

- Given labeled examples $\{(x_i, y_i)\}$

$$\mathbb{E}\left[y \cdot \mathcal{S}_m(x)\right] = \mathbb{E}\left[\nabla^{(m)} f(x)\right]$$

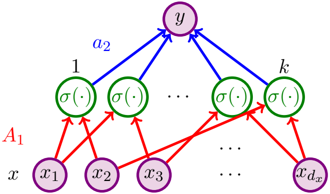
$$\Downarrow$$

$$\boxed{M_2 = \mathbb{E}[y \cdot \mathcal{S}_2(x)] = \sum_{j \in [k]} \lambda_{2,j} \cdot (A_1)_j \otimes (A_1)_j}$$



Moments of a Neural Network

$$\mathbb{E}[y|x] = f(x) = a_2^\top \sigma(A_1^\top x + b_1) + b_2$$

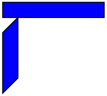
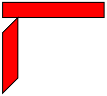


- Given labeled examples $\{(x_i, y_i)\}$

$$\mathbb{E}\left[y \cdot \mathcal{S}_m(x)\right] = \mathbb{E}\left[\nabla^{(m)} f(x)\right]$$

$$\Downarrow$$

$$M_2 = \mathbb{E}[y \cdot \mathcal{S}_2(x)] = \sum_{j \in [k]} \lambda_{2,j} \cdot (A_1)_j \otimes (A_1)_j$$

$=$  $+$  $\cdots\cdot$

$\lambda_{11}(A_1)_1 \otimes (A_1)_1$ $\lambda_{12}(A_1)_2 \otimes (A_1)_2$

Moments of a Neural Network

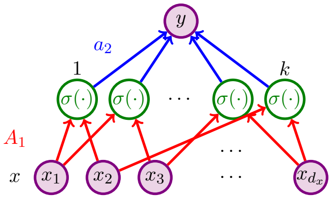
$$\boxed{\mathbb{E}[y|x] = f(x) = a_2^\top \sigma(A_1^\top x + b_1) + b_2}$$

- Given labeled examples $\{(x_i, y_i)\}$

$$\mathbb{E}\left[y \cdot \mathcal{S}_m(x)\right] = \mathbb{E}\left[\nabla^{(m)} f(x)\right]$$

$$\Downarrow$$

$$\boxed{M_3 = \mathbb{E}[y \cdot \mathcal{S}_3(x)] = \sum_{j \in [k]} \lambda_{3,j} \cdot (A_1)_j \otimes (A_1)_j \otimes (A_1)_j}$$



Moments of a Neural Network

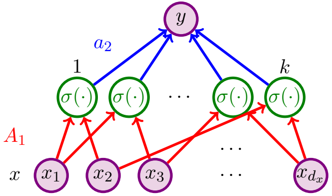
$$\boxed{\mathbb{E}[y|x] = f(x) = a_2^\top \sigma(A_1^\top x + b_1) + b_2}$$

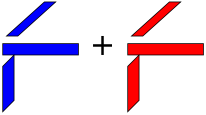
- Given labeled examples $\{(x_i, y_i)\}$

$$\mathbb{E}\left[y \cdot \mathcal{S}_m(x)\right] = \mathbb{E}\left[\nabla^{(m)} f(x)\right]$$

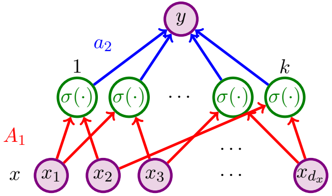
$$\Downarrow$$

$$\boxed{M_3 = \mathbb{E}[y \cdot \mathcal{S}_3(x)] = \sum_{j \in [k]} \lambda_{3,j} \cdot (A_1)_j \otimes (A_1)_j \otimes (A_1)_j}$$



=  ····

Moments of a Neural Network



$$\boxed{\mathbb{E}[y|x] = f(x) = a_2^\top \sigma(A_1^\top x + b_1) + b_2}$$

- Given labeled examples $\{(x_i, y_i)\}$

$$\mathbb{E}\left[y \cdot \mathcal{S}_m(x)\right] = \mathbb{E}\left[\nabla^{(m)} f(x)\right]$$

$$\Downarrow$$

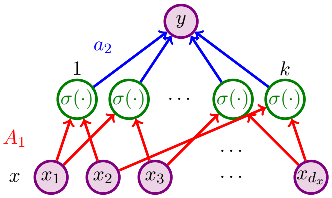
$$\boxed{M_3 = \mathbb{E}[y \cdot \mathcal{S}_3(x)] = \sum_{j \in [k]} \lambda_{3,j} \cdot (A_1)_j \otimes (A_1)_j \otimes (A_1)_j}$$

Why tensors are required?

- Matrix decomposition recovers subspace, not actual weights.
- Tensor decomposition uniquely recovers under non-degeneracy.

Moments of a Neural Network

$$\mathbb{E}[y|x] = f(x) = a_2^\top \sigma(A_1^\top x + b_1) + b_2$$



- Given labeled examples $\{(x_i, y_i)\}$

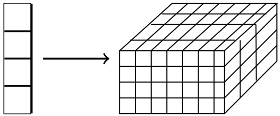
$$\mathbb{E}\left[y \cdot \mathcal{S}_m(x)\right] = \mathbb{E}\left[\nabla^{(m)} f(x)\right]$$

$$\Downarrow$$

$$M_3 = \mathbb{E}[y \cdot \mathcal{S}_3(x)] = \sum_{j \in [k]} \lambda_{3,j} \cdot (A_1)_j \otimes (A_1)_j \otimes (A_1)_j$$

- Guaranteed learning of weights of first layer via tensor decomposition.
- Learning the other parameters via a Fourier technique.

NN-LiFT: Neural Network LearnIng using Feature Tensors

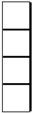


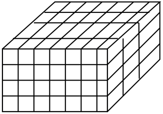
Input:
$x \in \mathbb{R}^d$

$\mathcal{S}_3(x) \in \mathbb{R}^{d\times d\times d}$

NN-LiFT: Neural Network LearnIng using Feature Tensors

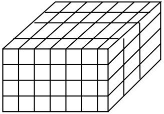
Estimating M_3 using
labeled data $\{(x_i, y_i)\}$





Cross-moment

$$\frac{1}{n}\sum_{i=1}^{n} y_i \otimes \mathcal{S}_3(x_i) = \frac{1}{n}\sum_{i=1}^{n} y_i \otimes$$



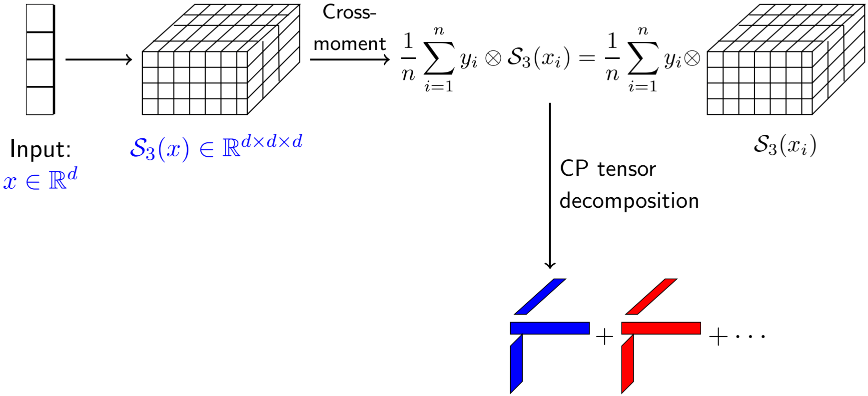
Input:
$x \in \mathbb{R}^d$

$\mathcal{S}_3(x) \in \mathbb{R}^{d\times d\times d}$

$\mathcal{S}_3(x_i)$

NN-LiFT: Neural Network LearnIng using Feature Tensors

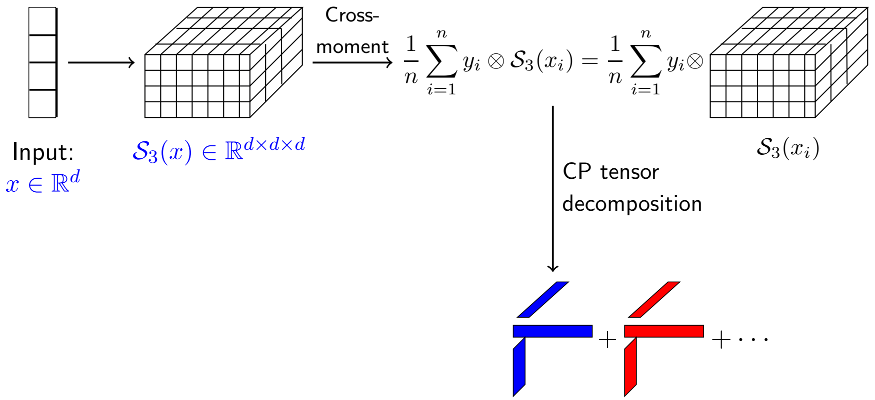
Estimating M_3 using labeled data $\{(x_i, y_i)\}$



Rank-1 components are the estimates of columns of A_1

NN-LiFT: Neural Network LearnIng using Feature Tensors

Estimating M_3 using labeled data $\{(x_i, y_i)\}$



Rank-1 components are the estimates of columns of A_1

Fourier technique $\Rightarrow a_2, b_1, b_2$

Estimation error bound

- Guaranteed learning of weights of first layer via tensor decomposition.

$$M_3 = \mathbb{E}[y \otimes \mathcal{S}_3(x)] = \sum_{j \in [k]} \lambda_{3,j} \cdot (A_1)_j \otimes (A_1)_j \otimes (A_1)_j$$

- Full column rank assumption on weight matrix A_1
- Guaranteed tensor decomposition (AGHKT'14, AGJ'14)

Estimation error bound

- Guaranteed learning of weights of first layer via tensor decomposition.

$$M_3 = \mathbb{E}[y \otimes \mathcal{S}_3(x)] = \sum_{j \in [k]} \lambda_{3,j} \cdot (A_1)_j \otimes (A_1)_j \otimes (A_1)_j$$

- Full column rank assumption on weight matrix A_1
- Guaranteed tensor decomposition (AGHKT'14, AGJ'14)
- Learning the other parameters via a Fourier technique.

Estimation error bound

- Guaranteed learning of weights of first layer via tensor decomposition.

$$M_3 = \mathbb{E}[y \otimes \mathcal{S}_3(x)] = \sum_{j \in [k]} \lambda_{3,j} \cdot (A_1)_j \otimes (A_1)_j \otimes (A_1)_j$$

- Full column rank assumption on weight matrix A_1
- Guaranteed tensor decomposition (AGHKT'14, AGJ'14)
- Learning the other parameters via a Fourier technique.

Theorem (JSA'14)

- number of samples $n = \mathrm{poly}(d, k)$, we have w.h.p.

$$|f(x) - \hat{f}(x)|^2 \leq \tilde{O}(1/n).$$

"Beating the Perils of Non-Convexity: Guaranteed Training of Neural Networks using Tensor Methods" by M. Janzamin, H. Sedghi and A., June. 2015.

Our Main Result: Risk Bounds

- Approximating arbitrary function $f(x)$ with bounded

$$C_f := \int_{\mathbb{R}^d} \|\omega\|_2 \cdot |F(\omega)| d\omega$$

- n samples, d input dimension, k number of neurons.

Our Main Result: Risk Bounds

- Approximating arbitrary function $f(x)$ with bounded

$$C_f := \int_{\mathbb{R}^d} \|\omega\|_2 \cdot |F(\omega)| d\omega$$

- n samples, d input dimension, k number of neurons.

Theorem(JSA'14)

- Assume C_f is small.

$$\mathbb{E}[|f(x) - \hat{f}(x)|^2] \leq O(C_f^2/k) + O(1/n).$$

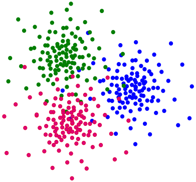
- Polynomial sample complexity n in terms of dimensions d, k.
- Computational complexity same as SGD with enough parallel processors.

"Beating the Perils of Non-Convexity: Guaranteed Training of Neural Networks using Tensor Methods" by M. Janzamin, H. Sedghi and A. , June. 2015.

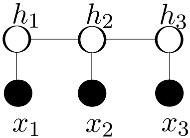
Outline

Tractable Learning for LVMs

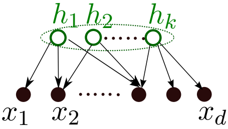
GMM



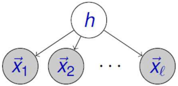
HMM



ICA



Multiview and Topic Models



$h \in [k]$,

$\vec{x}_1 \in \mathbb{R}^{d_1}, \vec{x}_2 \in \mathbb{R}^{d_2}, \ldots, \vec{x}_\ell \in \mathbb{R}^{d_\ell}$.

k = # components, ℓ = # views (*e.g.*, audio, video, text).

View 1: $\vec{x}_1 \in \mathbb{R}^{d_1}$

View 2: $\vec{x}_2 \in \mathbb{R}^{d_2}$

View 3: $\vec{x}_3 \in \mathbb{R}^{d_3}$

At Scale Tensor Computations

Randomized Tensor Sketches

- Naive computation scales exponentially in order of the tensor.
- Propose randomized FFT sketches.
- Computational complexity independent of tensor order.
- Linear scaling in input dimension and number of samples.

(1) *Fast and Guaranteed Tensor Decomposition via Sketching* by Yining Wang, Hsiao-Yu Tung, Alex Smola, A. , NIPS 2015.

(2) *Tensor Contractions with Extended BLAS Kernels on CPU and GPU* by Y. Shi, UN Niranjan, C. Cecka, A. Mowli, A.

At Scale Tensor Computations

Randomized Tensor Sketches

- Naive computation scales exponentially in order of the tensor.
- Propose randomized FFT sketches.
- Computational complexity independent of tensor order.
- Linear scaling in input dimension and number of samples.

Tensor Contractions with Extended BLAS Kernels on CPU and GPU

- BLAS: Basic Linear Algebraic Subprograms, highly optimized libraries.
- Use extended BLAS to minimize data permutation, I/O calls.

---

(1) *Fast and Guaranteed Tensor Decomposition via Sketching* by Yining Wang, Hsiao-Yu Tung, Alex Smola, A. , NIPS 2015.

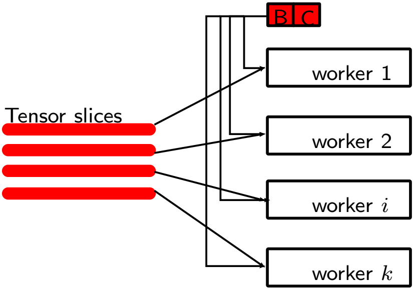
(2) *Tensor Contractions with Extended BLAS Kernels on CPU and GPU* by Y. Shi, UN Niranjan, C. Cecka, A. Mowli, A.

Preliminary Results on Spark

- In-memory processing of Spark: ideal for iterative tensor methods.
- Alternating Least Squares for Tensor Decomposition.

$$\min_{w,A,B,C} \left\| T - \sum_{i=1}^{k} \lambda_i A(:,i) \otimes B(:,i) \otimes C(:,i) \right\|_F^2$$

Update Rows Independently



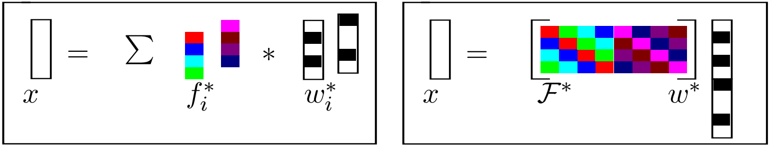
Results on NYtimes corpus

$3 * 10^5$ documents, 10^8 words

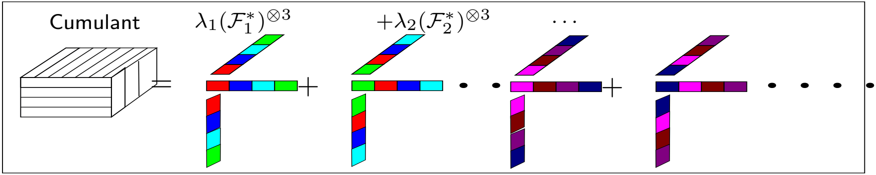
Spark	Map-Reduce
26mins	4 hrs

(2) *Topic Modeling at Lightning Speeds via Tensor Factorization on Spark* by F. Huang, A. , under preparation.

Convolutional Tensor Decomposition



(a) Convolutional dictionary model **(b)** Reformulated model



Efficient methods for tensor decomposition with circulant constraints.

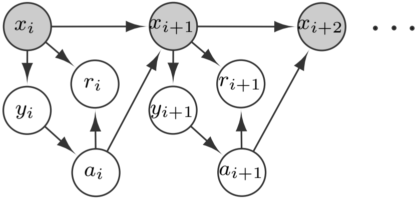
Convolutional Dictionary Learning through Tensor Factorization by F. Huang, A. , June 2015.

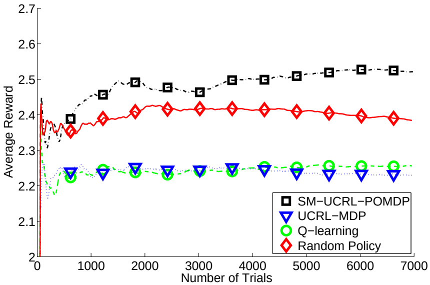
Reinforcement Learning (RL) of POMDPs

- Partially observable Markov decision processes.

Proposed Method

- Consider memoryless policies. Episodic learning: indirect exploration.
- Tensor methods: careful conditioning required for learning.
- First RL method for POMDPs with logarithmic regret bounds.





Logarithmic Regret Bounds for POMDPs using Spectral Methods by K. Azzizade, A. Lazaric, A. , under preparation.

Outline

Summary and Outlook

Summary

- Tensor methods: a powerful paradigm for guaranteed large-scale machine learning.
- First methods to provide provable bounds for training neural networks, many latent variable models (e.g HMM, LDA), POMDPs!

Summary and Outlook

Summary

- Tensor methods: a powerful paradigm for guaranteed large-scale machine learning.
- First methods to provide provable bounds for training neural networks, many latent variable models (e.g HMM, LDA), POMDPs!

Outlook

- Training multi-layer neural networks, models with invariances, reinforcement learning using neural networks ...
- Unified framework for tractable non-convex methods with guaranteed convergence to global optima?

My Research Group and Resources

- Podcast/lectures/papers/software available at
  http://newport.eecs.uci.edu/anandkumar/