Randomized Algorithms in Machine Learning

Tianbao Yang

Department of Computer Science
The University of Iowa

AMCS Seminar

April 24, 2015

Outline

Introduction

Machine Learning

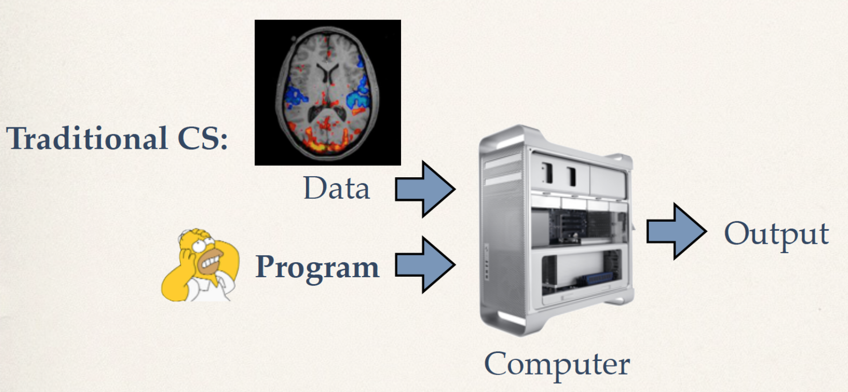
What is Machine Learning?

Arthur Samule (1959)

"Field of study that gives computers the ability to learn without being explicitly programmed"

Machine Learning

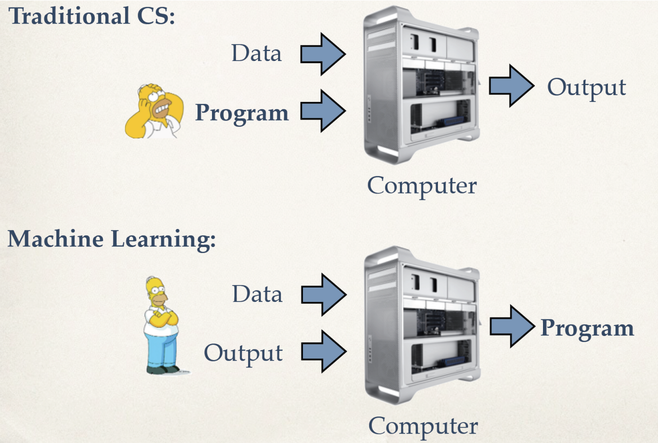
Traditional Computer Science



picture by courtesy of Killian Weinberger.

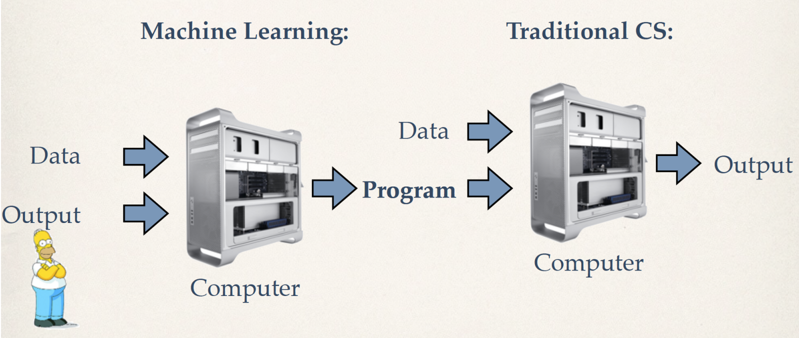
Machine Learning

Machine Learning vs. Traditional Computer Science



picture by courtesy of Killian Weinberger.

Machine Learning

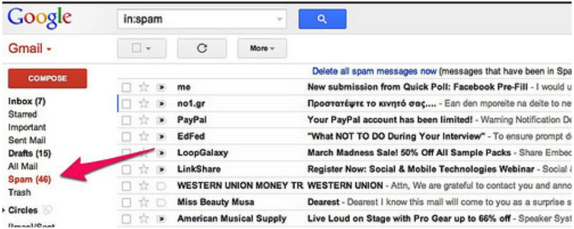


Let the Data Speak for itself!

picture by courtesy of Killian Weinberger.

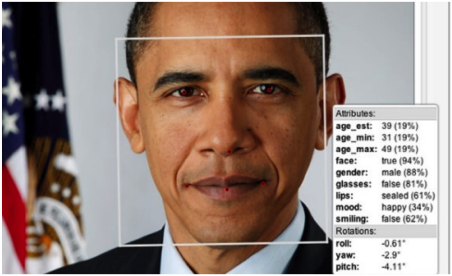
Applications of Machine Learning

Spam Filter



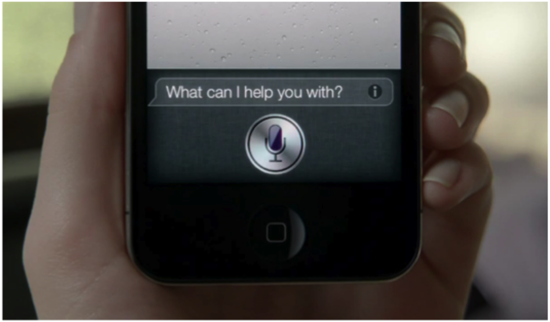
Applications of Machine Learning

Face Recognition

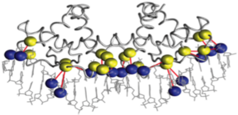


Applications of Machine Learning

Speech Recognition

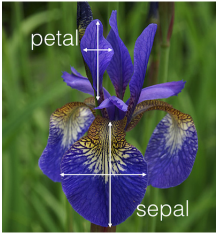


Applications of Machine Learning

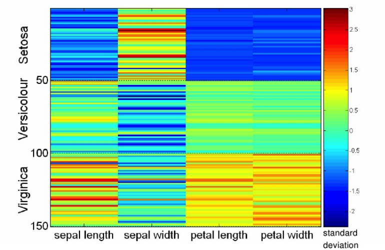


Data Matrices and Machine Learning

The Instance-feature Matrix: $X \in \mathbb{R}^{n \times d}$



$$X = \begin{pmatrix} \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \cdot \\ \cdot \\ \cdot \\ \mathbf{x}_n^\top \end{pmatrix}$$



Setosa

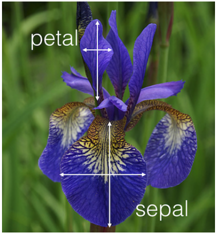
Versicolour

Virginica

Data Matrices and Machine Learning

The output vector: $\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ . \\ . \\ . \\ y_n \end{pmatrix}$

- continuous $y_i \in \mathbb{R}$: regression (e.g., house price)
- discrete, e.g., $y_i \in \{1, 2, 3\}$: classification (e.g., species of iris)



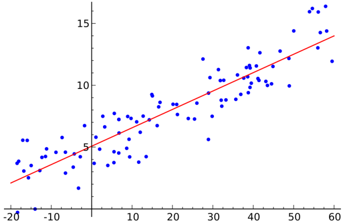
Data Matrices and Machine Learning

Many machine learning tasks are formulated based on the data matrix X and the output vector $\mathbf{y}$.

- Regression: (minimize the least-squares error)

$$\min_{\mathbf{w}\in\mathbb{R}^d} \|X\mathbf{w} - \mathbf{y}\|_2^2$$

$$= \min_{\mathbf{w}\in\mathbb{R}^d} \sum_{i=1}^{n} (\mathbf{x}_i^\top \mathbf{w} - y_i)^2$$



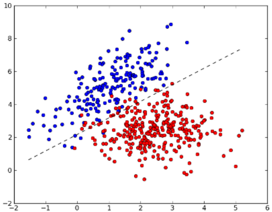
  - $\mathbf{w} \in \mathbb{R}^d$ refers to the predictive model (or the program as referred at the beginning)
  - Prediction on new data: $\mathbf{x}_{new}^\top \mathbf{w}_*$ ($\mathbf{w}_*$ optimizes the objective function)

Data Matrices and Machine Learning

Many machine learning tasks are formulated based on the data matrix X and the output vector $\mathbf{y}$.

- Classification

$$\min_{\mathbf{w}\in\mathbb{R}^d} \underbrace{\frac{1}{n}\sum_{i=1}^{n}\ell(\mathbf{w}^\top\mathbf{x}_i, y_i)}_{\text{Empirical Loss}} + \underbrace{\frac{\lambda}{2}\|\mathbf{w}\|_2^2}_{\textit{Regularization}}$$

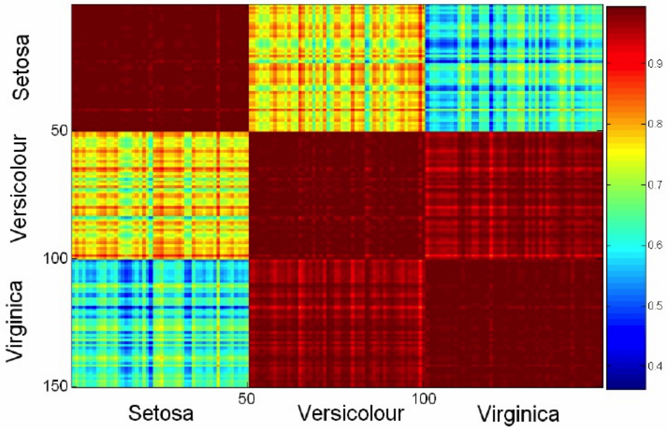


- e.g., $y_i \in \{1, -1\}$
- Loss function $\ell(z, y)$: $z = \mathbf{w}^\top\mathbf{x}$
  1. SVMs: (squared) hinge loss $\ell(z, y) = \max(0, 1 - yz)^p$, where $p = 1, 2$
  2. Logistic Regression: $\ell(z) = \log(1 + \exp(-yz))$

Data Matrices and Machine Learning

The Instance-Instance Matrix: $K \in \mathbb{R}^{n \times n}$

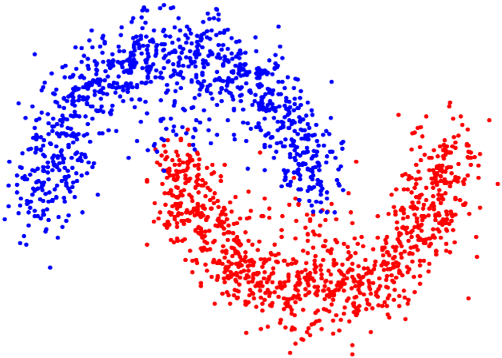
- Similarity Matrix
- Kernel Matrix



Data Matrices and Machine Learning

Some machine learning tasks are formulated on the kernel matrix

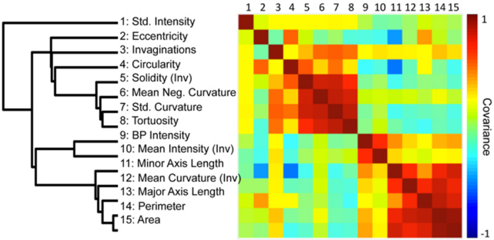
- Clustering
- Kernel Methods



Data Matrices and Machine Learning

The Feature-Feature Matrix: $C \in \mathbb{R}^{d \times d}$

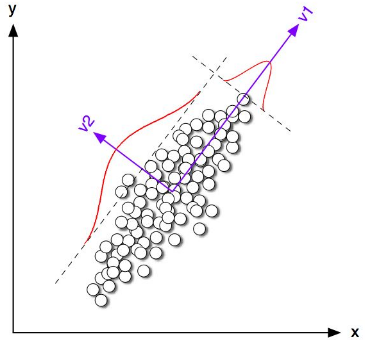
- Covariance Matrix
- Distance Metric Matrix



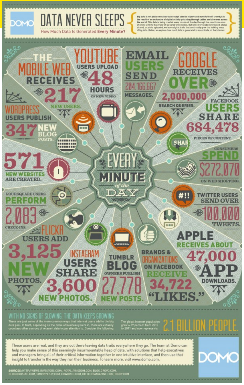
Data Matrices and Machine Learning

Some machine learning tasks requires the covariance matrix

- Principal Component Analysis
- Dimensionality Reduction
- Top-k Singular Value (Eigen-Value) Decomposition of the Covariance Matrix



Big Data Challenge



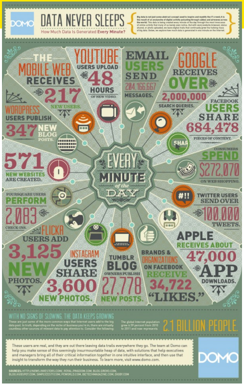
Huge amount of data generated every day

- Facebook users upload 3 million photos
- Goolge receives 3 billion queries
- Youtube users upload over 1,700 hours video
- Global internet population is 2.1 billion people
- 247 billion emails sent

http://www.visualnews.com/2012/06/19/how-much-data-created-every-minute/

Do we really need Big Data?

Big Data Challenge



Huge amount of data generated every day

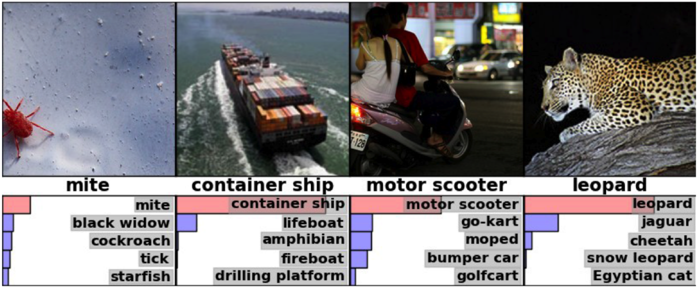
- Facebook users upload 3 million photos
- Goolge receives 3 billion queries
- Youtube users upload over 1,700 hours video
- Global internet population is 2.1 billion people
- 247 billion emails sent

http://www.visualnews.com/2012/06/19/how-much-data-created-every-minute/

Do we really need Big Data?

Big Data Challenge

General Visual Recognition Challenge (ImageNet Challenge)



Hundreds of Thousands of Objects

Big Data Challenge

Fine-grained Image Classification

(a) Siberian husky

(b) Eskimo dog

Big Data Challenge

Big Data will be the key to achieve success

Example: 1000 Objects Classification

- 14 millions of images indexed
- surpass human-level performance: top-1 accuracy 78% and top-5 accuracy 95%

Why Learning from Big Data is challenging?

Big Data Challenge

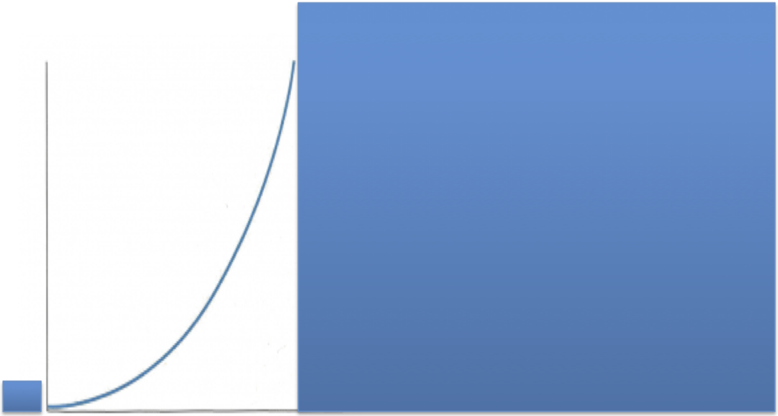
Big Data will be the key to achieve success

Example: 1000 Objects Classification

- 14 millions of images indexed
- surpass human-level performance: top-1 accuracy 78% and top-5 accuracy 95%

Why Learning from Big Data is challenging?

Why Big Data is challenging



Outline

Outline

Randomized Algorithms

- Use some kind of randomization (sampling) to reduce the cost of computation

Randomized Algorithms

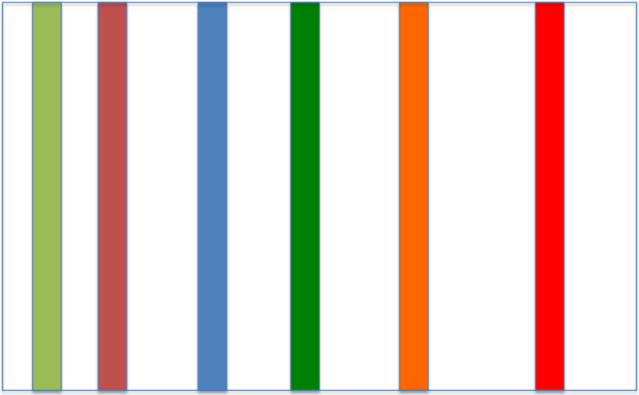
- Use some kind of randomization (sampling) to reduce the cost of computation

Randomized Algorithms

- Use some kind of randomization (sampling) to reduce the cost of computation (e.g., sampling rows or instances)

Randomized Algorithms

- Use some kind of randomization (sampling) to reduce the cost of computation (e.g., sampling columns or features)



Randomized Algorithms

Algorithms:

- Stochastic Optimization (e.g., SGD)
- Randomized Low-rank Matrix Approximation (e.g., randomized SVD)
- Dropout for Deep Learning
- Randomized reduction for regression and classification

Benefits:

- Faster
- More robust (implicit regularization)
- Easy to analyze
- exploit modern computational architectures

Randomized Algorithms

Algorithms:

- Stochastic Optimization (e.g., SGD)
- Randomized Low-rank Matrix Approximation (e.g., randomized SVD)
- Dropout for Deep Learning
- Randomized reduction for regression and classification

Benefits:

- Faster
- More robust (implicit regularization)
- Easy to analyze
- exploit modern computational architectures

Randomized Algorithms

Algorithms:

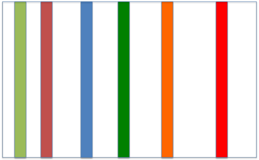
- Stochastic Optimization (e.g., SGD)
- Randomized Low-rank Matrix Approximation (e.g., randomized SVD)
- Dropout for Deep Learning
- Randomized reduction for regression and classification

Benefits:

- Faster
- More robust (implicit regularization)
- Easy to analyze
- exploit modern computational architectures

Randomized Feature Reduction for Classification

$$\min_{\mathbf{w}\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^{n} \ell(\mathbf{w}^\top \mathbf{x}_i, y_i) + \frac{\lambda}{2}\|\mathbf{w}\|_2^2$$

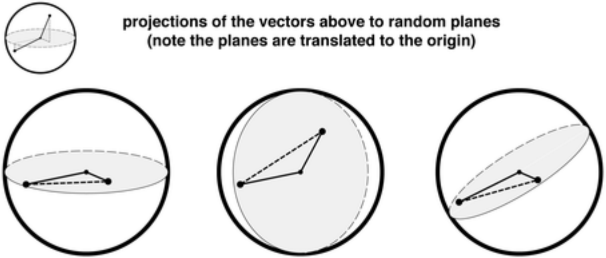


- Randomized feature reduction: $\widehat{\mathbf{x}}_i = A\mathbf{x}_i$, where $A \in \mathbb{R}^{m\times d}$ with $m \ll d$
- A: random projection matrix (e.g., Gaussian entries)
- Solve the reduced problem

$$\min_{\mathbf{u}\in\mathbb{R}^m} \frac{1}{n}\sum_{i=1}^{n} \ell(\mathbf{u}^\top \widehat{\mathbf{x}}_i, y_i) + \frac{\lambda}{2}\|\mathbf{u}\|_2^2$$

Why does Randomized Reduction Works?

The Johnson-Lindenstrauss Lemma (Johnson & Lindenstrauss (1984)).



Question

How can we recover a model in original high-dimensional space?

- Usually features in original feature space have meanings (e.g., genes, words)
- Finding a model in the original feature space can help understand the importance of different features
- Help us design better strategies (e.g., for controlling risk of a disease)

Question

How can we recover a model in original high-dimensional space?

- Usually features in original feature space have meanings (e.g., genes, words)
- Finding a model in the original feature space can help understand the importance of different features
- Help us design better strategies (e.g., for controlling risk of a disease)

Outline

Outline

Randomized Feature Reduction for Classification

$$\mathbf{w}_* = \arg\min_{\mathbf{w}\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^n \ell(\mathbf{w}^\top \mathbf{x}_i, y_i) + \frac{\lambda}{2}\|\mathbf{w}\|_2^2$$

- $\mathbf{x}_i \in \mathbb{R}^d$, expensive when d is very very large, e.g., millions or billions
- Randomized feature reduction: $\widehat{\mathbf{x}}_i = A\mathbf{x}_i$, where $A \in \mathbb{R}^{m\times d}$ with $m \ll d$
- Solve the reduced problem

$$\mathbf{u}_* = \arg\min_{\mathbf{u}\in\mathbb{R}^m} \frac{1}{n}\sum_{i=1}^n \ell(\mathbf{u}^\top \widehat{\mathbf{x}}_i, y_i) + \frac{\lambda}{2}\|\mathbf{u}\|_2^2$$

Question: How to obtain a good model $\widehat{\mathbf{w}}_*$ in the original feature space?

A Naive Approach

$$\mathbf{u}_* = \arg\min_{\mathbf{u}\in\mathbb{R}^m} \frac{1}{n}\sum_{i=1}^{n} \ell(\mathbf{u}^\top \widehat{\mathbf{x}}_i, y_i) + \frac{\lambda}{2}\|\mathbf{u}\|_2^2$$

$$\mathbf{u}_* = \arg\min_{\mathbf{u}\in\mathbb{R}^m} \frac{1}{n}\sum_{i=1}^{n} \ell(\mathbf{u}^\top A\mathbf{x}_i, y_i) + \frac{\lambda}{2}\|\mathbf{u}\|_2^2$$

Naive Recovery:

$$\widehat{\mathbf{w}}_* = A^\top \mathbf{u}_* \in \mathbb{R}^d$$

Problem: $\widehat{\mathbf{w}}_*$ could be a very bad solution

$$\|\widehat{\mathbf{w}}_* - \mathbf{w}_*\|_2 \geq \Omega\left(\sqrt{\frac{d-m}{d}}\|\mathbf{w}_*\|_2\right)$$

Dual Recovery

(COLT'13, IEEE-IT'14)

Our Approach: Dual Recovery

The Dual Problem: (using Fenchel conjugate)

$$\ell_i^*(\alpha_i) = \max_{\alpha_i} \alpha_i z - \ell(z, y_i)$$

$$\text{Primal } \mathbf{w}_* = \arg\min_{\mathbf{w}\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^n \ell(\mathbf{w}^\top \mathbf{x}_i, y_i) + \frac{\lambda}{2}\|\mathbf{w}\|_2^2$$

$$\text{Dual } \alpha_* = \arg\max_{\alpha\in\mathbb{R}^n} -\frac{1}{n}\sum_{i=1}^n \ell_i^*(\alpha_i) - \frac{1}{2\lambda n^2}\alpha^\top X X^\top \alpha$$

$$\mathbf{w}_* = -\frac{1}{\lambda n} X^\top \alpha_*$$

Our Approach: Dual Recovery

Important Implication from the Dual: $\mathbf{w}_*$ lies in the row space of the data matrix $X \in \mathbb{R}^{n\times d}$

- the Naive approach: $\widehat{\mathbf{w}}_* = A^\top \mathbf{u}_*$
- Dual Recovery: $\widetilde{\mathbf{w}}_* = -\frac{1}{\lambda n} X^\top \widehat{\alpha}_*$, where

$$\widehat{\alpha}_* = \arg\max_{\alpha\in\mathbb{R}^n} -\frac{1}{n}\sum_{i=1}^n \ell_i^*(\alpha_i) - \frac{1}{2\lambda n^2}\alpha^\top \widehat{X}\widehat{X}^\top \alpha$$

- $\widehat{X} = XA^\top \in \mathbb{R}^{n\times m}$
- Our theorem: under low-rank assumption of the data matrix X (e.g., $rank(X) = r$), with a high probability $1-\delta$,

$$\|\widetilde{\mathbf{w}}_* - \mathbf{w}_*\|_2 \leq \frac{\epsilon}{1-\epsilon}\|\mathbf{w}_*\|_2, \qquad \text{where } \epsilon = \Theta\left(\sqrt{\frac{r\log(r/\delta)}{m}}\right)$$

Our Approach: Dual Recovery

Important Implication from the Dual: $\mathbf{w}_*$ lies in the row space of the data matrix $X \in \mathbb{R}^{n\times d}$

- the Naive approach: $\widehat{\mathbf{w}}_* = A^\top \mathbf{u}_*$
- Dual Recovery: $\widetilde{\mathbf{w}}_* = -\frac{1}{\lambda n} X^\top \widehat{\alpha}_*$, where

$$\widehat{\alpha}_* = \arg\max_{\alpha\in\mathbb{R}^n} -\frac{1}{n}\sum_{i=1}^n \ell_i^*(\alpha_i) - \frac{1}{2\lambda n^2}\alpha^\top \widehat{X}\widehat{X}^\top \alpha$$

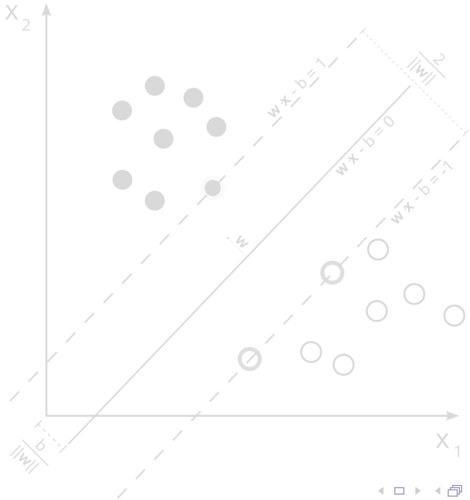
- $\widehat{X} = XA^\top \in \mathbb{R}^{n\times m}$
- Our theorem: under low-rank assumption of the data matrix X (e.g., $rank(X) = r$), with a high probability $1-\delta$,

$$\|\widetilde{\mathbf{w}}_* - \mathbf{w}_*\|_2 \leq \frac{\epsilon}{1-\epsilon}\|\mathbf{w}_*\|_2, \quad \text{where } \epsilon = \Theta\left(\sqrt{\frac{r\log(r/\delta)}{m}}\right)$$

Can you remove low-rank assumption?

Yes, we can. How?

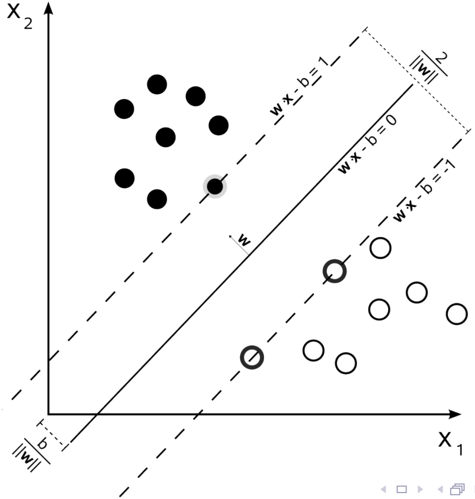
by exploiting the sparsity of the dual variables



Can you remove low-rank assumption?

Yes, we can. How?

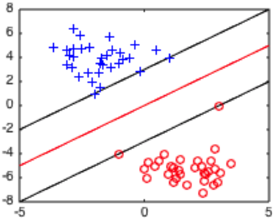
by exploiting the sparsity of the dual variables



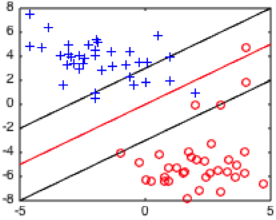
Dual-sparse Recovery

(To appear in ICML'15)

Can you remove low-rank assumption?



High-dimensional Space



low-dimensional Space

Our New Approach: Dual-sparse Recovery

- Dual-sparse Recovery: $\widetilde{\mathbf{w}}_* = -\frac{1}{\lambda n} X^\top \widehat{\alpha}_*$, where

$$\widehat{\alpha}_* = \arg\max_{\alpha \in \mathbb{R}^n} -\frac{1}{n}\sum_{i=1}^n \ell_i^*(\alpha_i) - \frac{1}{2\lambda n^2}\alpha^\top \widehat{X}\widehat{X}^\top \alpha - \frac{\tau}{n}\|\alpha\|_1$$

- Our theorem: if α_* is s-sparse, with a high probability $1-\delta$,

$$\|\widetilde{\mathbf{w}}_* - \mathbf{w}_*\|_2 \leq \epsilon \|\mathbf{w}_*\|_2, \quad \text{where } \epsilon = \Theta\left(\sqrt{\frac{s\log(n/\delta)}{m}}\right)$$

- Exploit Convex Optimization theory, JL lemma, Compressive Sensing theory

Our New Approach: Dual-sparse Recovery

- Dual-sparse Recovery: $\widetilde{\mathbf{w}}_* = -\frac{1}{\lambda n} X^\top \widehat{\alpha}_*$, where

$$\widehat{\alpha}_* = \arg\max_{\alpha\in\mathbb{R}^n} -\frac{1}{n}\sum_{i=1}^{n} \ell_i^*(\alpha_i) - \frac{1}{2\lambda n^2}\alpha^\top \widehat{X}\widehat{X}^\top \alpha - \frac{\tau}{n}\|\alpha\|_1$$

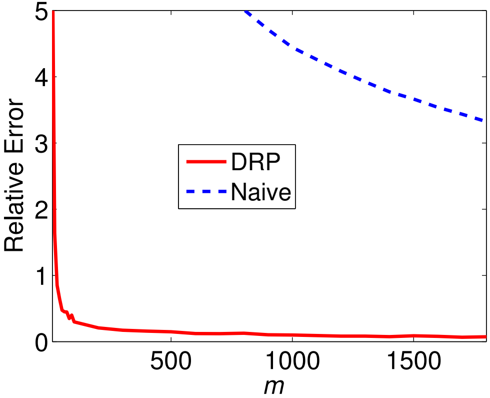
- Our theorem: if α_* is s-sparse, with a high probability $1-\delta$,

$$\|\widetilde{\mathbf{w}}_* - \mathbf{w}_*\|_2 \leq \epsilon \|\mathbf{w}_*\|_2, \quad \text{where } \epsilon = \Theta\left(\sqrt{\frac{s\log(n/\delta)}{m}}\right)$$

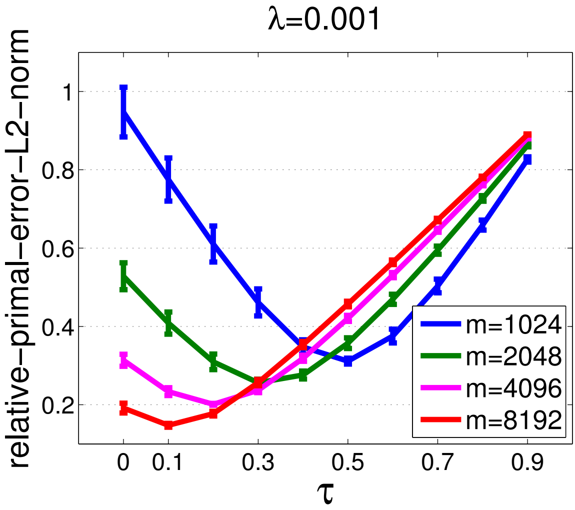
- Exploit Convex Optimization theory, JL lemma, Compressive Sensing theory

Results

Dual Recovery vs Naive Recovery

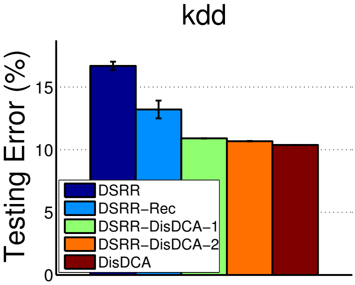


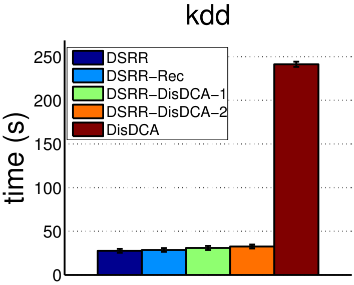
Dual-sparse Recovery



Big Data Experiments

KDDcup Data: $n = 8,407,752$, $d = 29,890,095$, 10 machines





Outline

Outline

Messages

- Machine Learning is changing our life
- Machine Learning is not just about Programming
- Big Data brings ground-breaking advances
- Randomized Algorithms are useful for Big Data
- If you are interested in any of these topics, I am happy to discuss with you.

Thank You!

Randomized Algorithms for Optimization

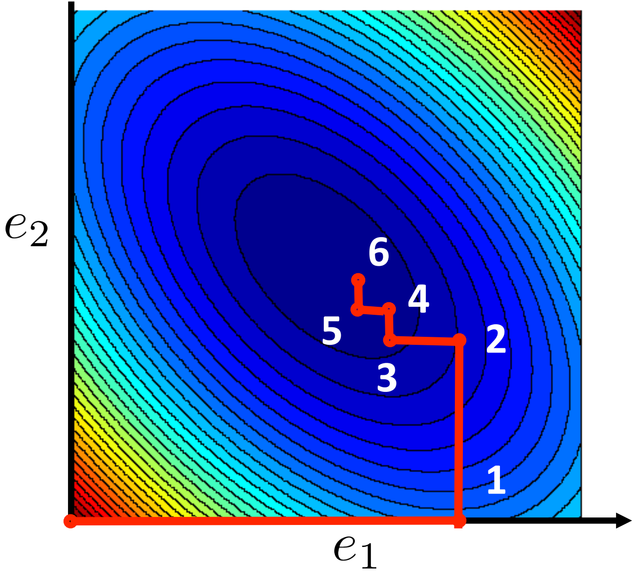
Stochastic Gradient Descent in Machine Learning

$$F(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} \ell(\mathbf{w}^\top \mathbf{x}_i, y_i) + \frac{\lambda}{2}\|\mathbf{w}\|_2^2$$

- let $i_t \in \{1, \ldots, n\}$ uniformly randomly sampled
- key equation: $\mathrm{E}_{i_t}[\nabla \ell(\mathbf{w}^\top \mathbf{x}_{i_t}, y_{i_t}) + \lambda \mathbf{w}] = \nabla F(\mathbf{w})$
- computation is cheaper $O(d)$ compared with full gradient $O(nd)$

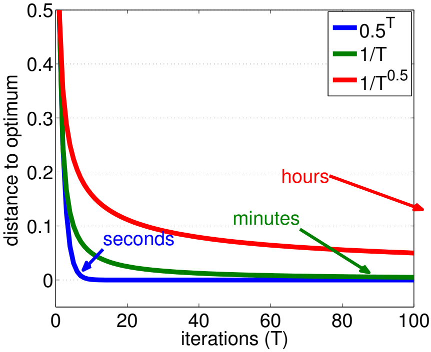
$$\mathbf{w}_t = (1 - \gamma_t \lambda)\mathbf{w}_{t-1} - \gamma_t \nabla \ell(\mathbf{w}_{t-1}^\top \mathbf{x}_{i_t}, y_{i_t})$$

Stochastic Coordinate Descent



Research on Stochastic Optimization

- Establish Fast Convergence Rate for various learning problems.
- Convex Optimization Theory
- Our Research
  - SGD with only one projection for complex domains (NIPS'12)
  - Distributed Stochastic Dual Coordinate Ascent (NIPS'13)

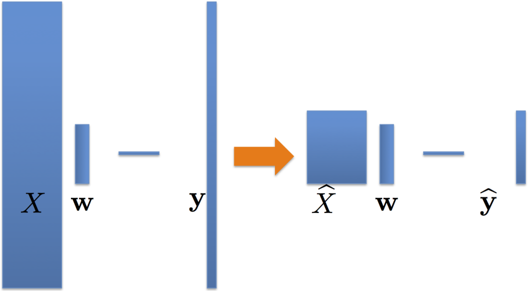


Randomized Reduction Methods

Over-constrained Least Squares Regression (LSR)

$$\min_{\mathbf{w}\in\mathbb{R}^d} \|X\mathbf{w}-\mathbf{y}\|_2, \quad where \quad X\in\mathbb{R}^{n\times d}, n \gg d$$

- Randomized Reduction $A \in \mathbb{R}^{m\times n} : \mathbb{R}^n \to \mathbb{R}^m$, $m \ll n$
- $\min_{\mathbf{w}\in\mathbb{R}^d} \|(AX)\mathbf{w} - (A\mathbf{y})\|_2$
- Time complexity: $O(nd^2) \to o(nd^2)$
- Mahoney (2011)



Research on Randomized Over-constrained LSR

$$\mathbf{w}_* = \arg\min_{\mathbf{w}\in\mathbb{R}^d} \|X\mathbf{w} - \mathbf{y}\|_2$$
$$\widehat{\mathbf{w}}_* = \arg\min_{\mathbf{w}\in\mathbb{R}^d} \|(AX)\mathbf{w} - (A\mathbf{y})\|_2$$

- What is a appropriate reduction matrix $A \in \mathbb{R}^{m\times n}$?
- The error bound of $\|\widehat{\mathbf{w}}_* - \mathbf{w}_*\|_2$
- Convex optimization theory, random matrix theory
- Our Research
  - A New Sampling Distribution for A (to appear in ICML'15)

Randomized Algorithms for Low-rank Matrix Approximation

low-rank matrix approximation

Many machine learning problems require computing the top-k components of the singular value decomposition (SVD)

- Principal Component Analysis
- Latent Semantic Indexing (information retrieval)

Given a $m \times n$ large matrix, how to efficiently compute its top-k components (SVD)?

RA for low-rank matrix approximation

Traditional Methods

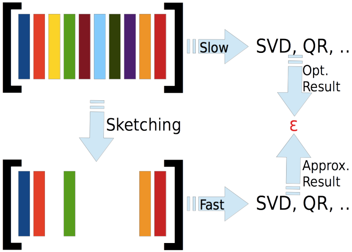
- SVD: $O(\min(mn^2, m^2 n))$
- partial SVD (for top-k components): $O(mnk)$
- rank revealing QR factorization: $O(mnk)$

Randomized Algorithms Halko et al. (2011)

- more robust
- can be as fast as $O(mn \log(k))$

RA for low-rank matrix approximation

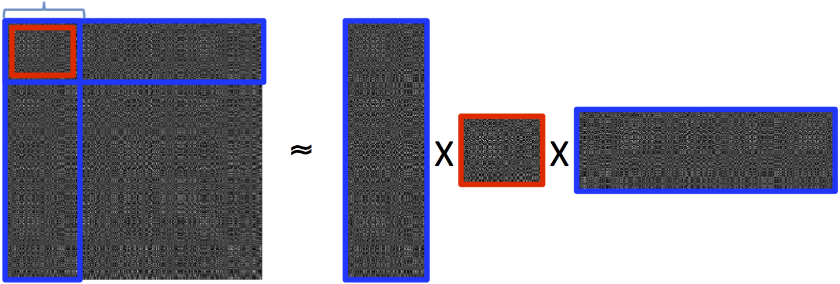
Random Sketching



- Random Projection: $\Omega \in \mathbb{R}^{n\times\ell}$, $\ell > k$ (random projection or random fourier transform); compute $B = A\Omega \in \mathbb{R}^{m\times\ell}$; compute the top-$k$ components based on B
- Column Subset Selection (CSS): sample a subset of columns
- CUR decomposition: $X = CUR$, sample columns and rows

CUR decomposition for Kernel matrix

the Nyström method



Research on RA Low-rank Martrix Approximation

The relative error of the approximated low-rank matrix

$$\|X - \hat{X}_k\|_{2,F} \leq (1+\epsilon)\|X - X_k\|_{2,F}$$

- Our Research
  - Better Bounds on the Nyström method (NIPS'12, IEEE-IT)
  - Better Sampling Distributions for CSS (to appear in ICML'15).

Why low-rank assumption?

$$\alpha_* = \arg\max_{\alpha\in\mathbb{R}^n} -\frac{1}{n}\sum_{i=1}^n \ell_i^*(\alpha_i) - \frac{1}{2\lambda n^2}\alpha^\top XX^\top \alpha$$

$$\widehat{\alpha}_* = \arg\max_{\alpha\in\mathbb{R}^n} -\frac{1}{n}\sum_{i=1}^n \ell_i^*(\alpha_i) - \frac{1}{2\lambda n^2}\alpha^\top XA^\top AX^\top \alpha$$

Let $X = U\Sigma V^\top$: $V \in \mathbb{R}^{d\times r}$

$$U\Sigma \underbrace{V^\top A^\top A V}_{BB^\top} \Sigma U^\top, \quad U\Sigma \underbrace{V^\top V}_{I_r} \Sigma U^\top$$

$B \in \mathbb{R}^{r\times m}$ tail bounds for the eigenvalues of a sum of random matrices

$$\|BB^\top - I\|_2 \leq O\left(\sqrt{\frac{r}{m}}\right)$$

Why low-rank assumption?

$$\alpha_* = \arg\max_{\alpha \in \mathbb{R}^n} -\frac{1}{n}\sum_{i=1}^{n} \ell_i^*(\alpha_i) - \frac{1}{2\lambda n^2}\alpha^\top XX^\top \alpha$$

$$\widehat{\alpha}_* = \arg\max_{\alpha \in \mathbb{R}^n} -\frac{1}{n}\sum_{i=1}^{n} \ell_i^*(\alpha_i) - \frac{1}{2\lambda n^2}\alpha^\top XA^\top AX^\top \alpha$$

Let $X = U\Sigma V^\top$: $V \in \mathbb{R}^{d\times r}$

$$U\Sigma \underbrace{V^\top A^\top A V}_{BB^\top} \Sigma U^\top, \quad U\Sigma \underbrace{V^\top V}_{I_r} \Sigma U^\top$$

$B \in \mathbb{R}^{r\times m}$ tail bounds for the eigenvalues of a sum of random matrices

$$\|BB^\top - I\|_2 \leq O\left(\sqrt{\frac{r}{m}}\right)$$

Why low-rank assumption?

$$\alpha_* = \arg\max_{\alpha \in \mathbb{R}^n} -\frac{1}{n}\sum_{i=1}^{n} \ell_i^*(\alpha_i) - \frac{1}{2\lambda n^2}\alpha^\top X X^\top \alpha$$

$$\widehat{\alpha}_* = \arg\max_{\alpha \in \mathbb{R}^n} -\frac{1}{n}\sum_{i=1}^{n} \ell_i^*(\alpha_i) - \frac{1}{2\lambda n^2}\alpha^\top X A^\top A X^\top \alpha$$

Let $X = U\Sigma V^\top$: $V \in \mathbb{R}^{d\times r}$

$$U\Sigma \underbrace{V^\top A^\top A V}_{BB^\top} \Sigma U^\top, \quad U\Sigma \underbrace{V^\top V}_{I_r} \Sigma U^\top$$

$B \in \mathbb{R}^{r\times m}$ tail bounds for the eigenvalues of a sum of random matrices

$$\|BB^\top - I\|_2 \le O\left(\sqrt{\frac{r}{m}}\right)$$

References I

Halko, N., Martinsson, P. G., and Tropp, J. A. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 53:217–288, 2011.

Johnson, William and Lindenstrauss, Joram. Extensions of Lipschitz mappings into a Hilbert space. In *Conference in modern analysis and probability (New Haven, Conn., 1982)*, volume 26, pp. 189–206. 1984.

Mahoney, Michael W. Randomized algorithms for matrices and data. *Foundations and Trends in Machine Learning*, 3:123–224, 2011.