High-dimensional prediction: Some computational challenges

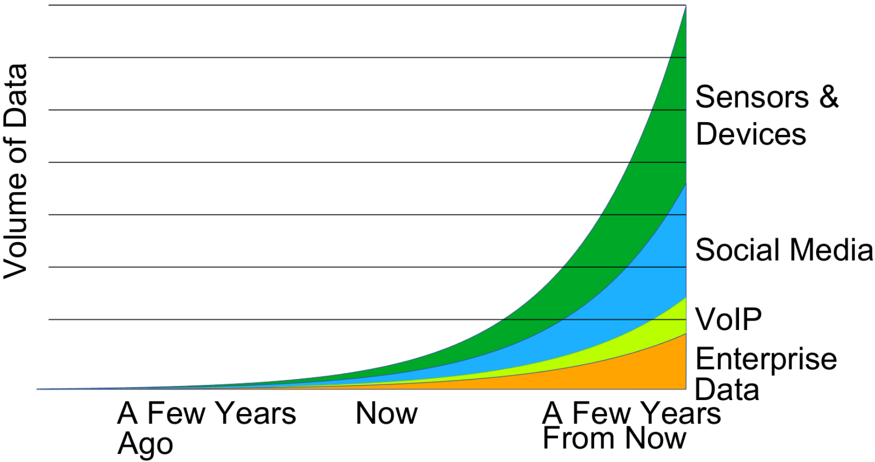
Martin Wainwright

UC Berkeley
Statistics and EECS

Baker-Kingland Lecture
Predictive Inference and its Applications

Joint work with: Mert Pilanci, Univ. Michigan
Yuting Wei, Carnegie Mellon University
Fanny Yang, ETH Zurich

The Data Explosion



- Every day: 2.5 billion gigabytes of data created
- Last two years: creation of 90% of the world's data (source: IBM)
- Data stored grows 4X faster than world economy
  (source: Mayer-Schonberger)

A few inconvenient truths...

- Even "simple" prediction problems can become computationally challenging.

A few inconvenient truths...

- Even "simple" prediction problems can become computationally challenging.

- "Naive" data storage often impossible $\Longrightarrow$ dimensionality reduction and distributed methods are needed.

A few inconvenient truths...

- Even “simple” prediction problems can become computationally challenging.

- “Naive” data storage often impossible $\Longrightarrow$ dimensionality reduction and distributed methods are needed.

- Computational considerations need to be considered jointly with statistical ones.

A few inconvenient truths...

- Even “simple” prediction problems can become computationally challenging.

- “Naive” data storage often impossible $\Longrightarrow$ dimensionality reduction and distributed methods are needed.

- Computational considerations need to be considered jointly with statistical ones.

- Interesting trade-offs between computational and statistical efficiency.

A few inconvenient truths...

- Even "simple" prediction problems can become computationally challenging.
- "Naive" data storage often impossible $\Longrightarrow$ dimensionality reduction and distributed methods are needed.
- Computational considerations need to be considered jointly with statistical ones.
- Interesting trade-offs between computational and statistical efficiency.

Today's talk: Two vignettes

§1 Data sketches: randomized dimensionality reduction

§2 Early stopping of iterative algorithms for prediction

§. 1. Randomized sketches of data

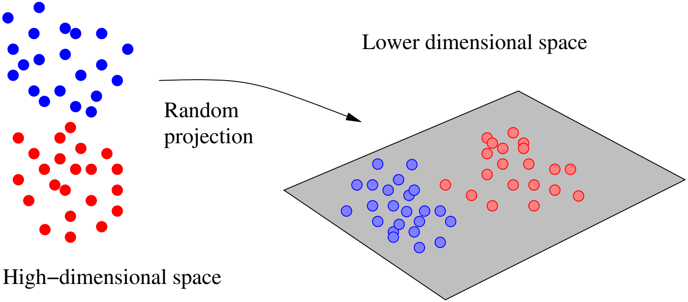
Massive data sets require fast algorithms but with rigorous guarantees.

§. 1. Randomized sketches of data

Massive data sets require fast algorithms but with rigorous guarantees.

Randomized projection is a general purpose tool:

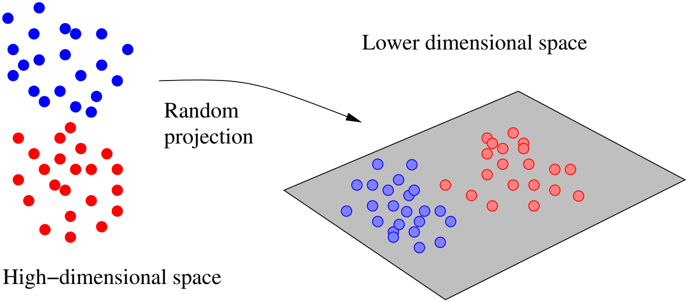
- Choose a random subspace of "low" dimension m.
- Project data into subspace, and solve reduced dimension problem.



§. 1. Randomized sketches of data

Randomized projection is a general purpose tool:

- Choose a random subspace of "low" dimension m.
- Project data into subspace, and solve reduced dimension problem.



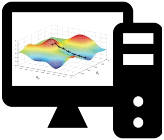
Widely studied and used:

- Johnson & Lindenstrauss (1984): in Banach/Hilbert space geometry
- various surveys and books: Vempala, 2004; Mahoney et al., 2011 Cormode et al., 2012.

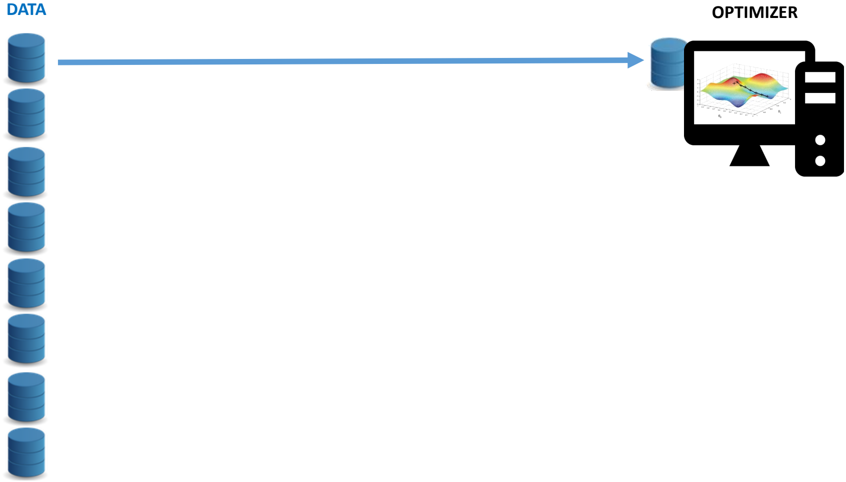
Randomized sketches for statistical optimization

DATA

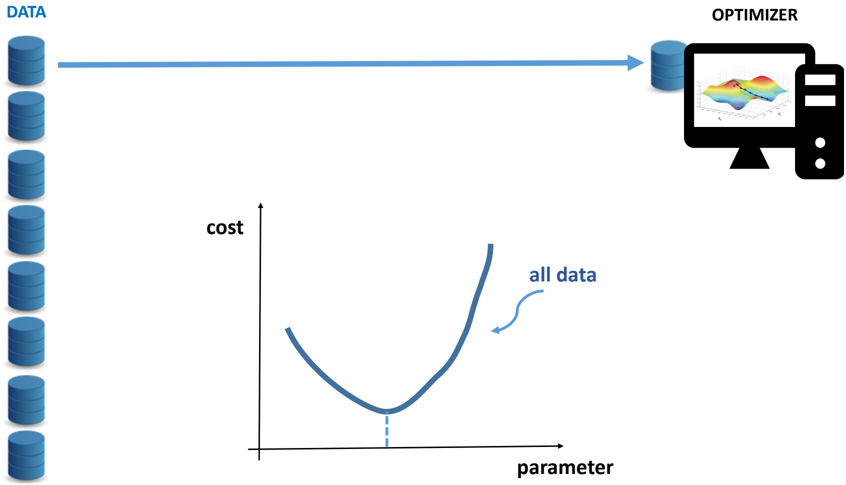
OPTIMIZER



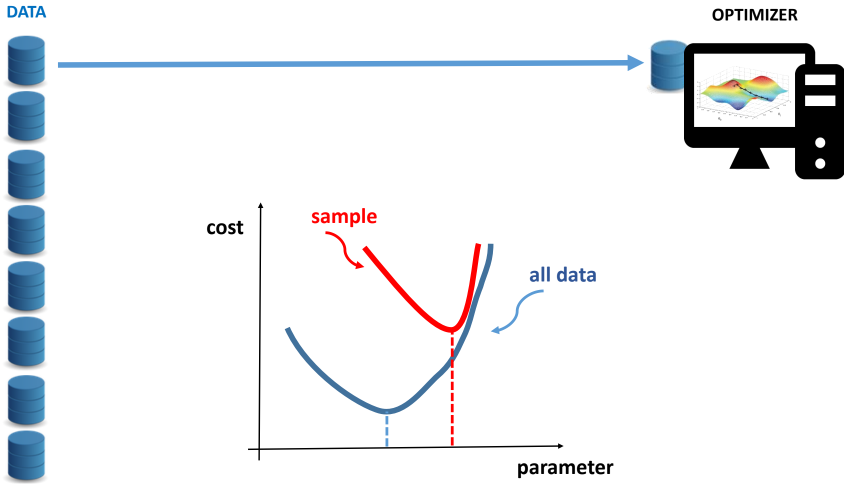
Randomized sketches for statistical optimization



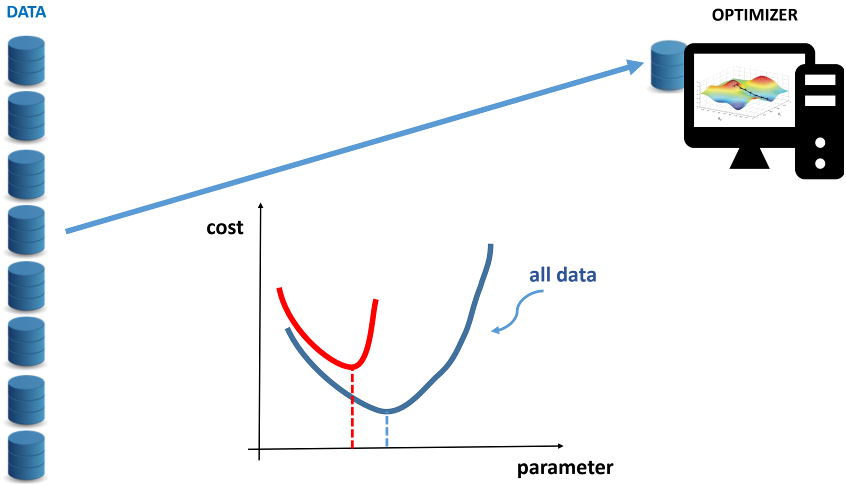
Randomized sketches for statistical optimization



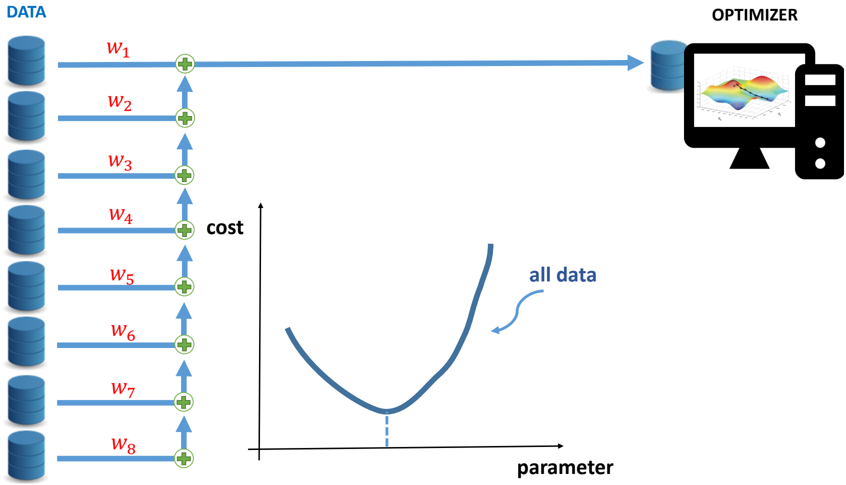
Randomized sketches for statistical optimization



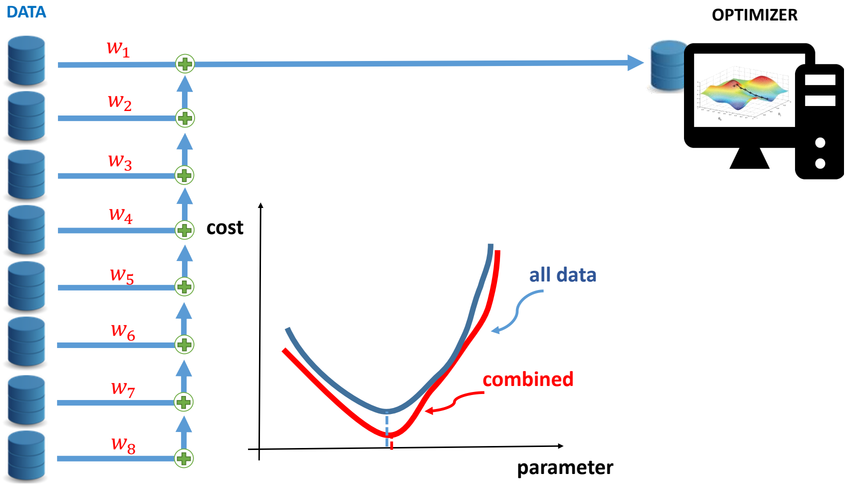
Randomized sketches for statistical optimization



Randomized sketches for statistical optimization



Randomized sketches for statistical optimization



Randomized projection for constrained least-squares

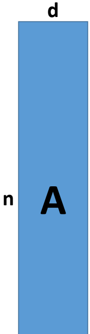
- Given data matrix $A \in \mathbb{R}^{n \times d}$, and response vector $y \in \mathbb{R}^n$
- Least-squares over convex constraint set $\mathcal{C} \subseteq \mathbb{R}^d$:

$$x_{\text{LS}} = \arg\min_{x \in \mathcal{C}} \underbrace{\|Ax - y\|_2^2}_{f(Ax)}$$

Randomized projection for constrained least-squares

- Given data matrix $A \in \mathbb{R}^{n \times d}$, and response vector $y \in \mathbb{R}^n$
- Least-squares over convex constraint set $\mathcal{C} \subseteq \mathbb{R}^d$:

$$x_{\text{LS}} = \arg\min_{x \in \mathcal{C}} \underbrace{\|Ax - y\|_2^2}_{f(Ax)}$$



Randomized projection for constrained least-squares

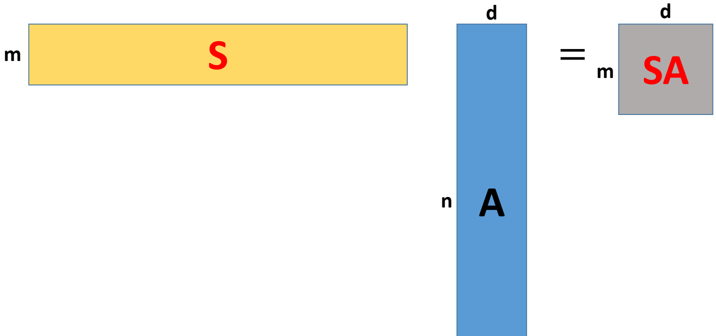
- Given data matrix $A \in \mathbb{R}^{n \times d}$, and response vector $y \in \mathbb{R}^n$
- Least-squares over convex constraint set $\mathcal{C} \subseteq \mathbb{R}^d$:

$$x_{\text{LS}} = \arg\min_{x \in \mathcal{C}} \underbrace{\|Ax - y\|_2^2}_{f(Ax)}$$

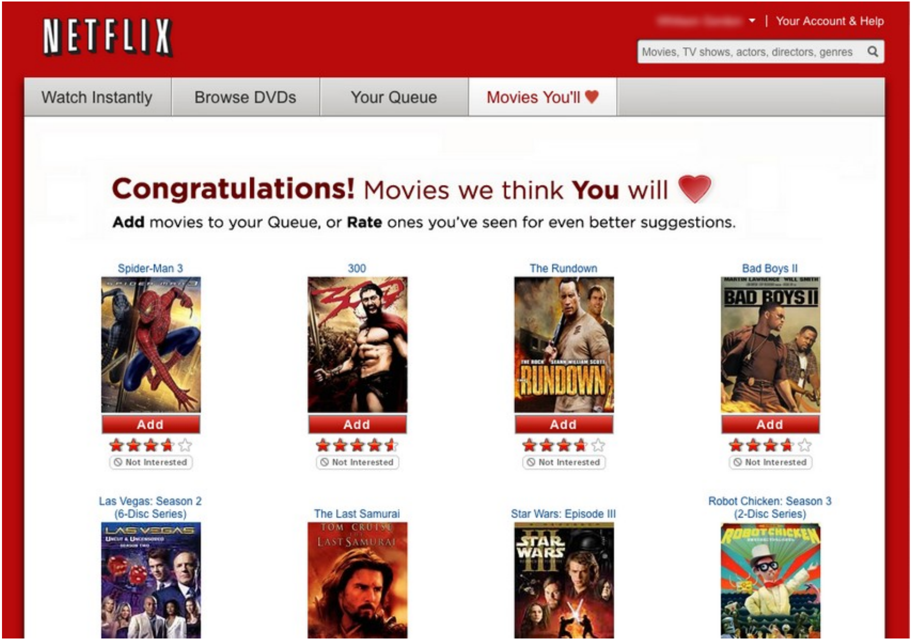
- Randomized approximation: (Sarlos, 2006, Mahoney et al., 2011)

$$\widehat{x} = \arg\min_{x \in \mathcal{C}} \|S(Ax - y)\|_2^2$$

- Random projection matrix $S \in \mathbb{R}^{m \times n}$



Application to Netflix data



Netflix data set

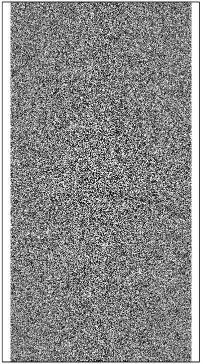
- 2 million $\times$ 17000 matrix A of ratings (users $\times$ movies)
- Predict the ratings of a particular movie
- Least-squares regression with ℓ_2 regularization

$$\min_{x \in \mathbb{R}^{17000}} \left\{ \|Ax - y\|_2^2 + \lambda \|x\|_2^2 \right\}$$

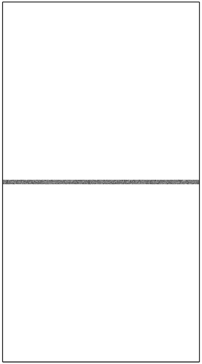
- Partition into test and training sets, solve for all values of $\lambda \in \{1, 2, ..., 100\}$.

Sketching for Netflix movie database

- original data set: 2 million $\times$ 17000 matrix A of ratings (users $\times$ movies)
- perform sketching (randomized dimensionality reduction)



Original data matrix



Sketched data matrix

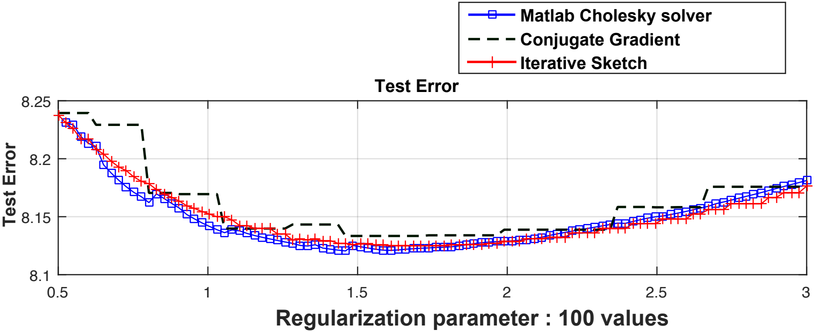
Key fact:

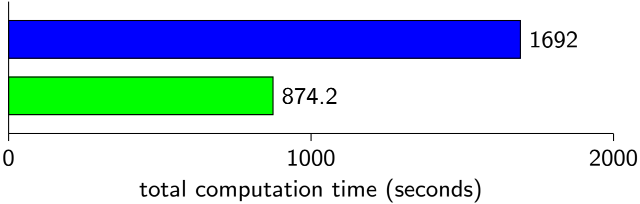
Sketching to dimension 2000 is enough!

Sketch is 2000 $\times$ 17000 : a few Megabytes.

Fitting the full regularization path

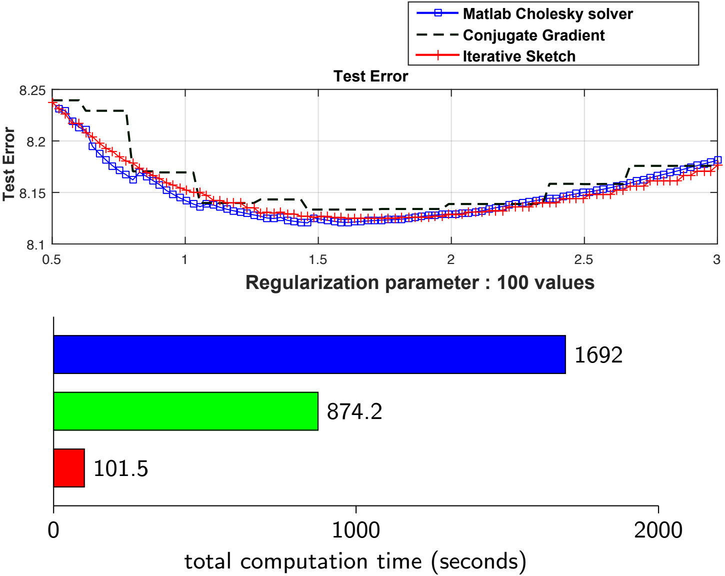
(Pilanci & W., 2016, J. Machine Learning Research)



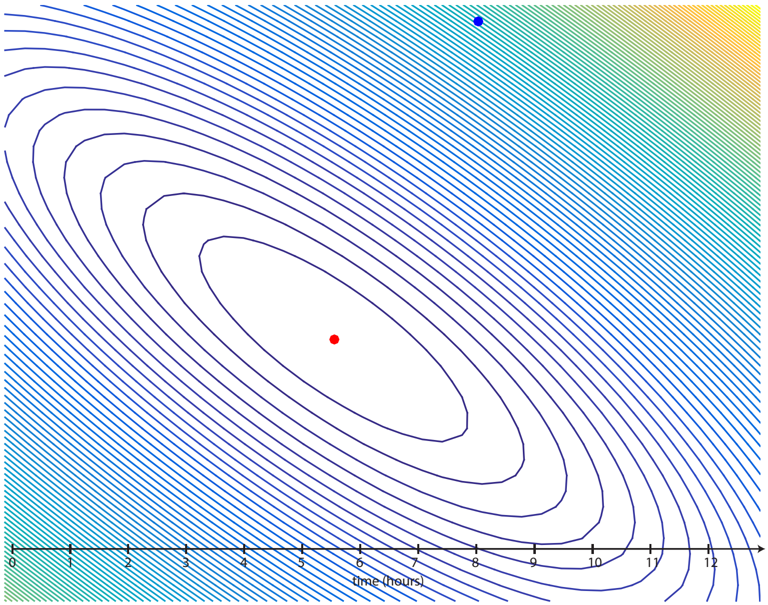


Fitting the full regularization path

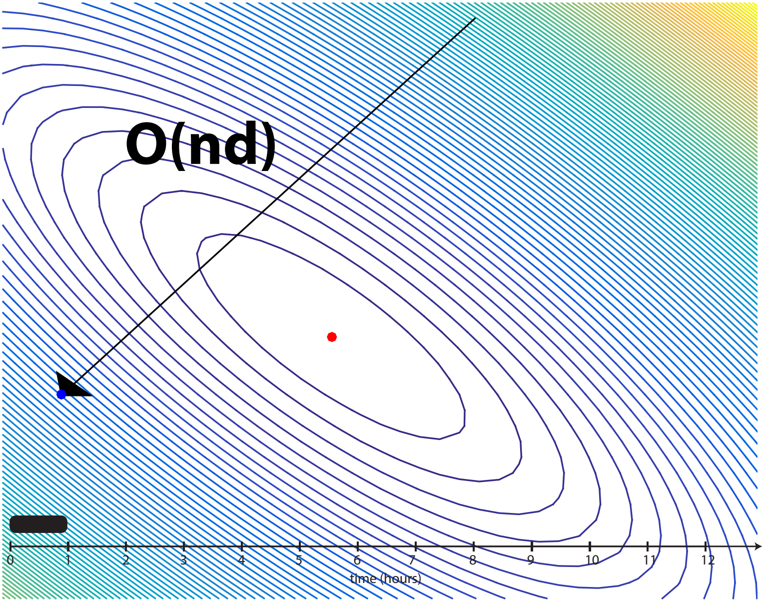
(Pilanci & W., 2016, J. Machine Learning Research)



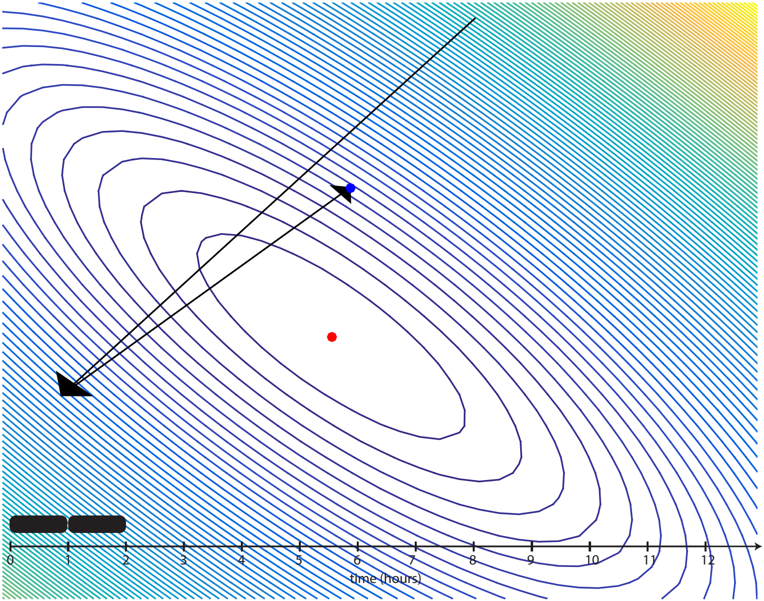
Gradient Descent



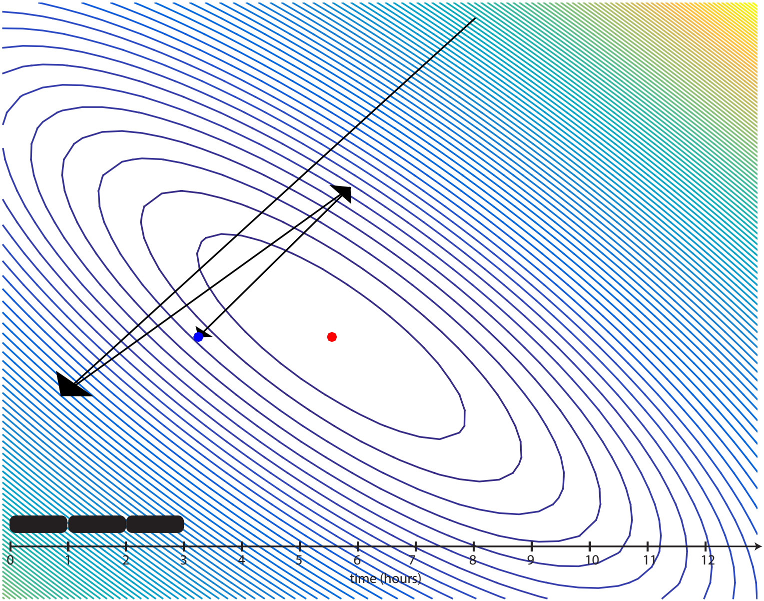
Gradient Descent



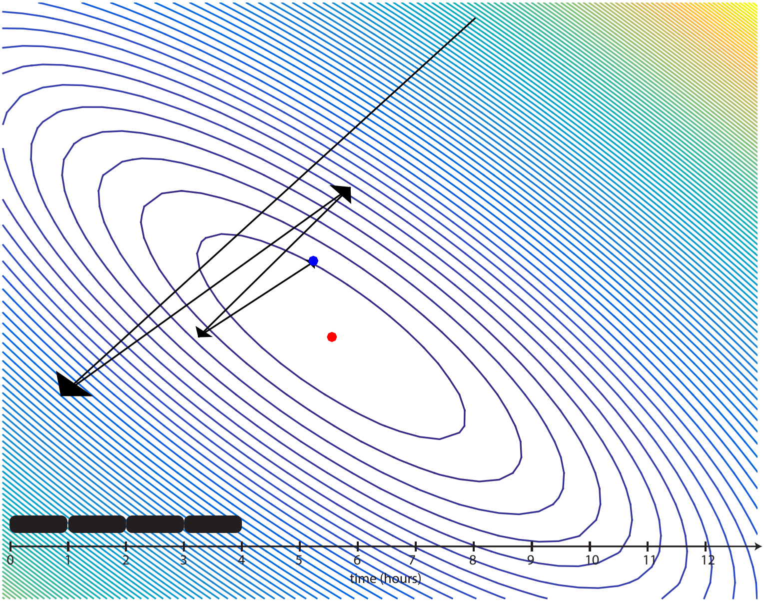
Gradient Descent



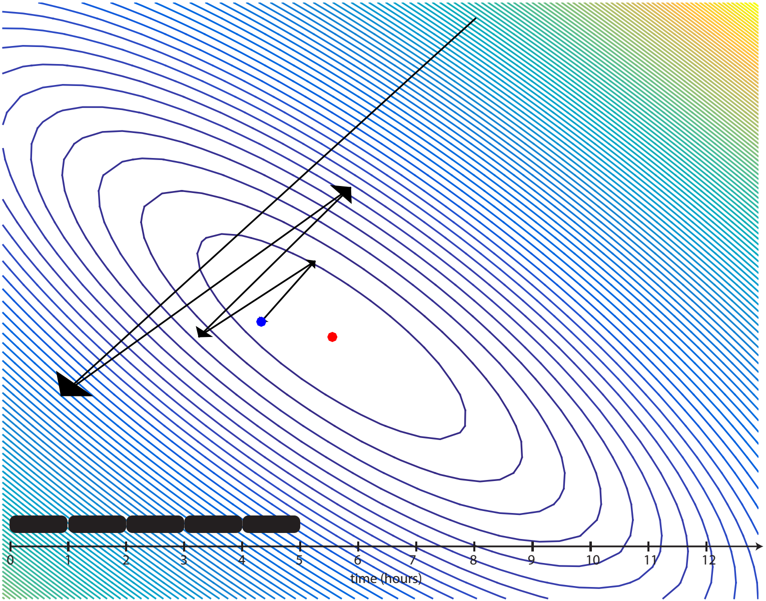
Gradient Descent



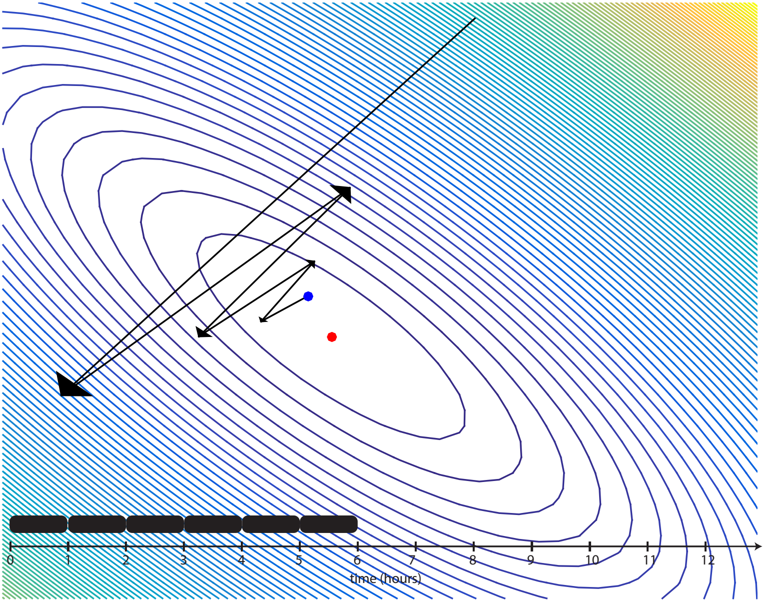
Gradient Descent



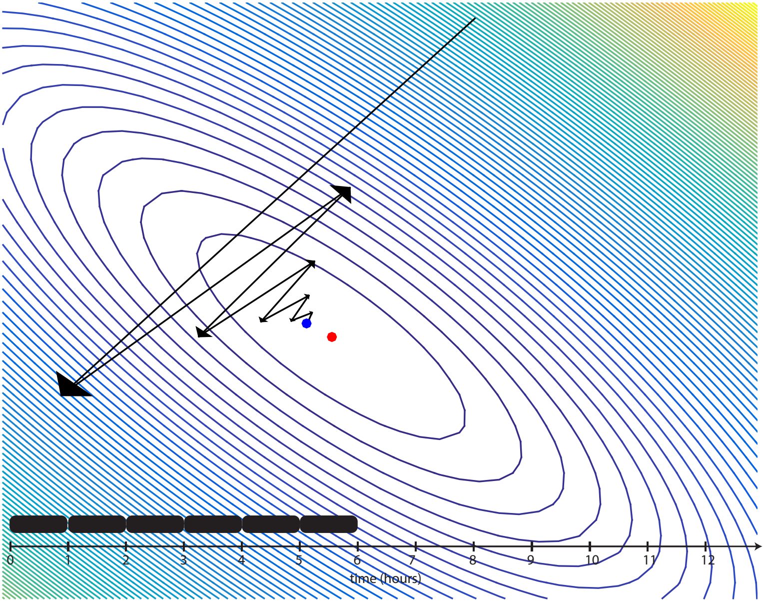
Gradient Descent



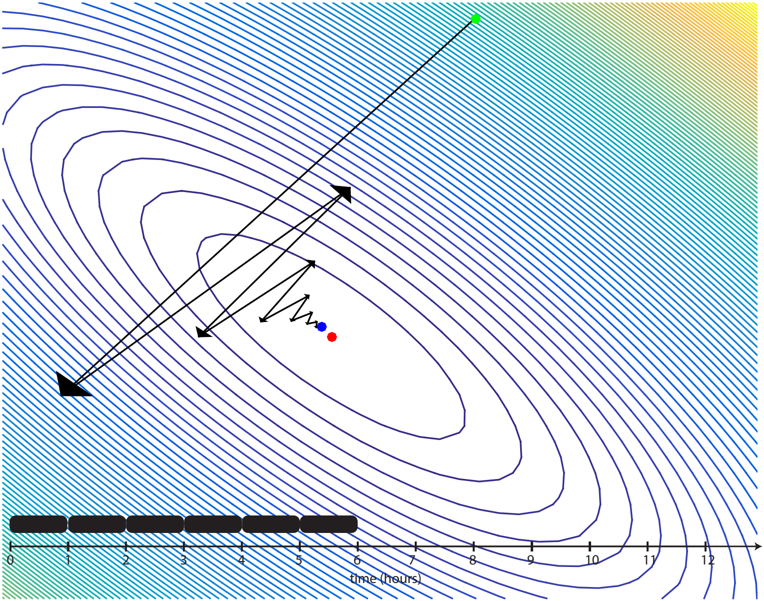
Gradient Descent



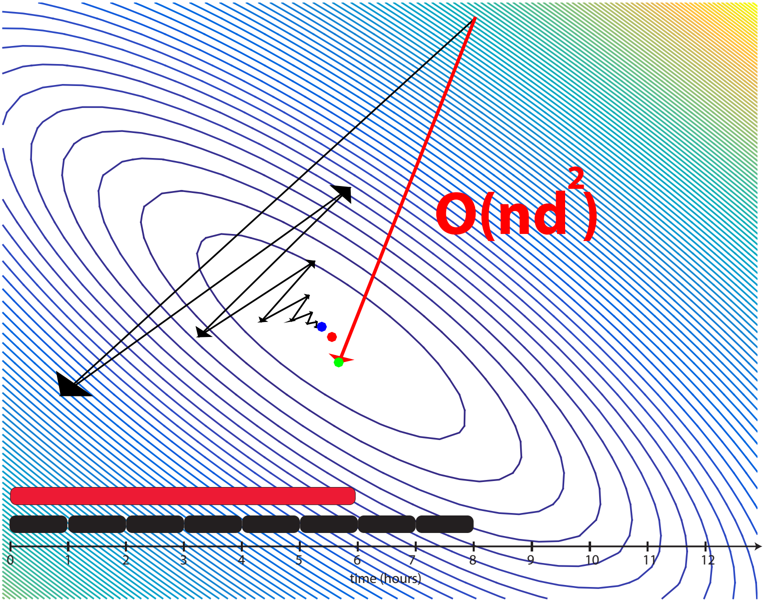
Gradient Descent vs



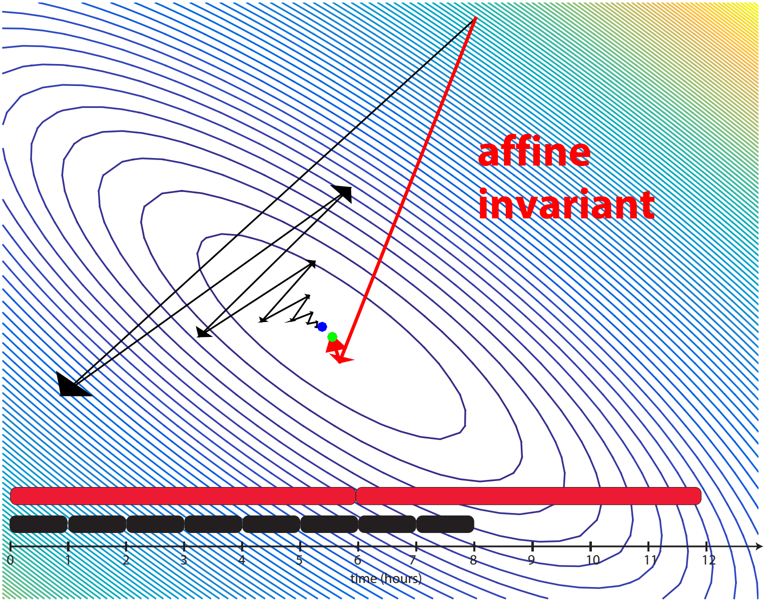
Gradient Descent vs Newton's Method



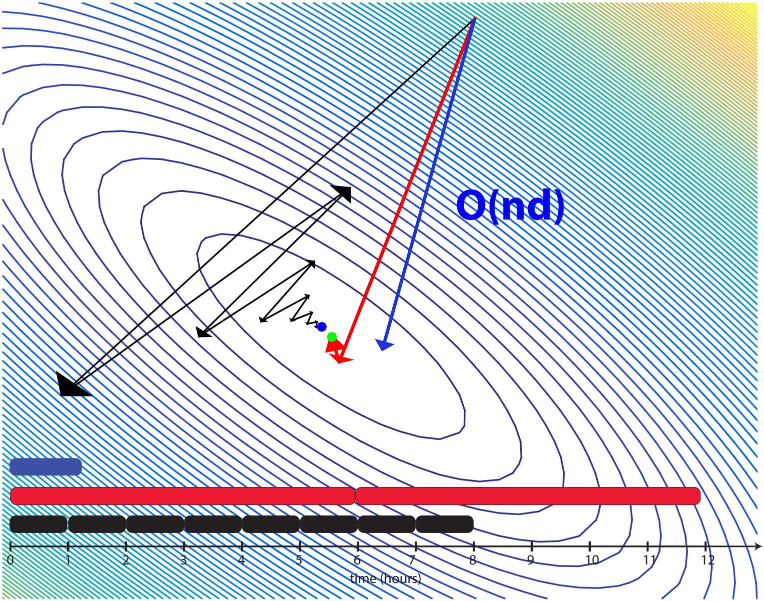
Gradient Descent vs Newton's Method



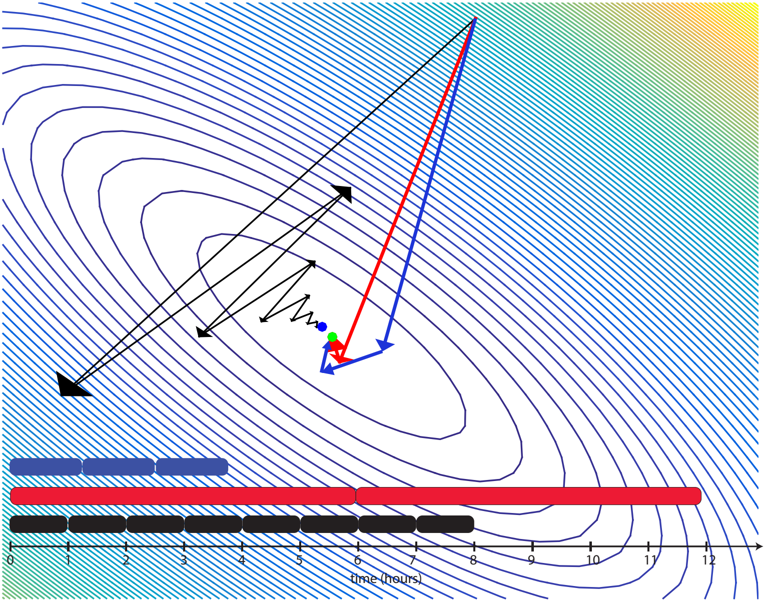
Gradient Descent vs Newton's Method



Gradient Descent vs Newton's Method



Gradient Descent vs Newton's Method



Iterative sketching for general data-based objectives

Goal: Minimize $g(x) = f(Ax)$ over convex set $\mathcal{C} \subseteq \mathbb{R}^d$:

$$x_{\text{opt}} = \arg\min_{x \in \mathcal{C}} g(x), \quad \text{where } g : \mathbb{R}^d \to \mathbb{R} \text{ is twice-differentiable.}$$

Iterative sketching for general data-based objectives

Goal: Minimize $g(x) = f(Ax)$ over convex set $\mathcal{C} \subseteq \mathbb{R}^d$:

$$x_{\text{opt}} = \arg\min_{x \in \mathcal{C}} g(x), \quad \text{where } g : \mathbb{R}^d \to \mathbb{R} \text{ is twice-differentiable.}$$

Ordinary Newton steps:

$$x^{t+1} = \arg\min_{x \in \mathcal{C}} \Big\{ \frac{1}{2} \|\nabla^2 g(x^t)^{1/2}(x - x^t)\|_2^2 + \langle \nabla g(x^t), \, x - x^t \rangle \Big\},$$

where $\nabla^2 g(x^t)^{1/2}$ is matrix square root Hessian at x^t.
Cost per step: $\mathcal{O}(nd^2)$ in unconstrained case.

Iterative sketching for general data-based objectives

Goal: Minimize $g(x) = f(Ax)$ over convex set $\mathcal{C} \subseteq \mathbb{R}^d$:

$$x_{\text{opt}} = \arg\min_{x \in \mathcal{C}} g(x), \quad \text{where } g : \mathbb{R}^d \to \mathbb{R} \text{ is twice-differentiable.}$$

Ordinary Newton steps:

$$x^{t+1} = \arg\min_{x \in \mathcal{C}} \Big\{ \frac{1}{2} \|\nabla^2 g(x^t)^{1/2}(x - x^t)\|_2^2 + \langle \nabla g(x^t), \, x - x^t \rangle \Big\},$$

where $\nabla^2 g(x^t)^{1/2}$ is matrix square root Hessian at x^t.
Cost per step: $\mathcal{O}(nd^2)$ in unconstrained case.

Sketched Newton steps: Using random sketch matrix S^t:

$$\tilde{x}^{t+1} = \arg\min_{x \in \mathcal{C}} \Big\{ \frac{1}{2} \|S^t \nabla^2 g(x^t)^{1/2}(x - \tilde{x}^t)\|_2^2 + \langle \nabla g(\tilde{x}^t), \, x - \tilde{x}^t \rangle \Big\}.$$

Cost per step: $\widetilde{\mathcal{O}}(nd)$ in unconstrained case.

Convergence of Newton sketch

Run algorithm with sketch dimension $m \asymp d$ on a self-concordant function $g(x) = f(Ax)$, and data matrix $A \in \mathbb{R}^{n \times d}$ with $n \gg d$.

Convergence of Newton sketch

Run algorithm with sketch dimension $m \asymp d$ on a self-concordant function $g(x) = f(Ax)$, and data matrix $A \in \mathbb{R}^{n \times d}$ with $n \gg d$.

Theorem (Pilanci & W, SIAM J. Opt, 2017)

With probability at least $1 - c_0 e^{-c_1 m}$, number of iterations required for ϵ accuracy is less than

$$c_2 \log(1/\epsilon)$$

where (c_0, c_1, c_2) are universal (problem-independent) constants.

Convergence of Newton sketch

Run algorithm with sketch dimension $m \asymp d$ on a self-concordant function $g(x) = f(Ax)$, and data matrix $A \in \mathbb{R}^{n \times d}$ with $n \gg d$.

Theorem (Pilanci & W, SIAM J. Opt, 2017)

With probability at least $1 - c_0 e^{-c_1 m}$, number of iterations required for ϵ accuracy is less than

$$c_2 \log(1/\epsilon)$$

where (c_0, c_1, c_2) are universal (problem-independent) constants.

Dependence on sample size n, dimension d; conditioning κ; and tolerance ϵ

Algorithm	Computational cost
Gradient Descent	$\mathcal{O}(\kappa\, n\, d \log(1/\epsilon))$
Acc. gradient Descent	$\mathcal{O}(\sqrt{\kappa}\, nd \log(1/\epsilon))$
Newton's Method	$\mathcal{O}(nd^2 \log\log(1/\epsilon))$
Newton Sketch	$\widetilde{\mathcal{O}}(nd \log(1/\epsilon))$

Convergence of Newton sketch

Run algorithm with sketch dimension $m \asymp d$ on a self-concordant function $g(x) = f(Ax)$, and data matrix $A \in \mathbb{R}^{n \times d}$ with $n \gg d$.

Theorem (Pilanci & W, SIAM J. Opt, 2017)

With probability at least $1 - c_0 e^{-c_1 m}$, number of iterations required for ϵ accuracy is less than

$$c_2 \log(1/\epsilon)$$

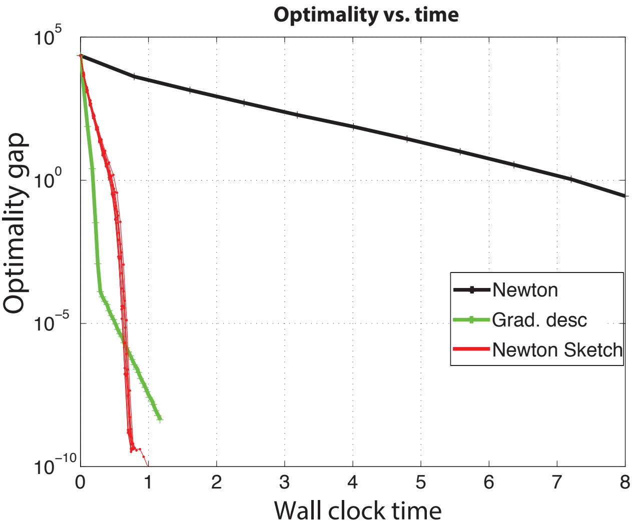
where (c_0, c_1, c_2) are universal (problem-independent) constants.

Dependence on sample size n, dimension d; conditioning κ; and tolerance ϵ

Algorithm	Computational cost
Gradient Descent	$\mathcal{O}(\kappa\, n\, d \log(1/\epsilon))$
Acc. gradient Descent	$\mathcal{O}(\sqrt{\kappa}\, nd \log(1/\epsilon))$
Newton's Method	$\mathcal{O}(nd^2 \log\log(1/\epsilon))$
Newton Sketch	$\widetilde{\mathcal{O}}(nd \log(1/\epsilon))$

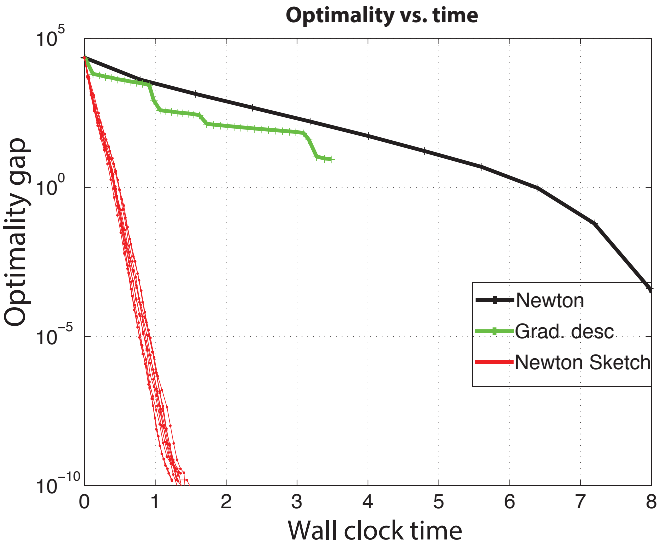
Note: Dependence on condition number κ unavoidable among 1st-order methods (Nesterov, 2004)

Logistic regression: uncorrelated features



Sample size $n = 500,000$ with $d = 5,000$ features

Logistic regression: correlated features



Sample size $n = 500,000$ with $d = 5,000$ features

§. 2. Non-parametric regression via boosting

Non-parametric regression problem: approximate the regression function $f^*(x) = \mathbb{E}[Y \mid X = x]$ based on samples $\{(x_i, y_i)\}_{i=1}^n$.

§. 2. Non-parametric regression via boosting

Non-parametric regression problem: approximate the regression function $f^*(x) = \mathbb{E}[Y \mid X = x]$ based on samples $\{(x_i, y_i)\}_{i=1}^n$.

Empirical loss function	Function class $\mathscr{F}$
$\mathcal{L}_n : \mathscr{F} \to \mathbb{R}$	Norm $\| \|_{\mathscr{F}}$

Given step sizes $\alpha^t > 0$:

$$f^{t+1} = f^t - \alpha^t g^t \qquad \text{where } g^t = \arg \max_{\|g\|_{\mathscr{F}} \leq 1} \langle g, \nabla \mathcal{L}_n(f^t) \rangle$$

[Freund & Schapire, 1997; Mason et al., 1999; Friedman et al., 2000]

§. 2. Non-parametric regression via boosting

Non-parametric regression problem: approximate the regression function $f^*(x) = \mathbb{E}[Y \mid X = x]$ based on samples $\{(x_i, y_i)\}_{i=1}^n$.

Empirical loss function	Function class $\mathscr{F}$
$\mathcal{L}_n : \mathscr{F} \to \mathbb{R}$	Norm $\| \, \|_{\mathscr{F}}$

Given step sizes $\alpha^t > 0$:

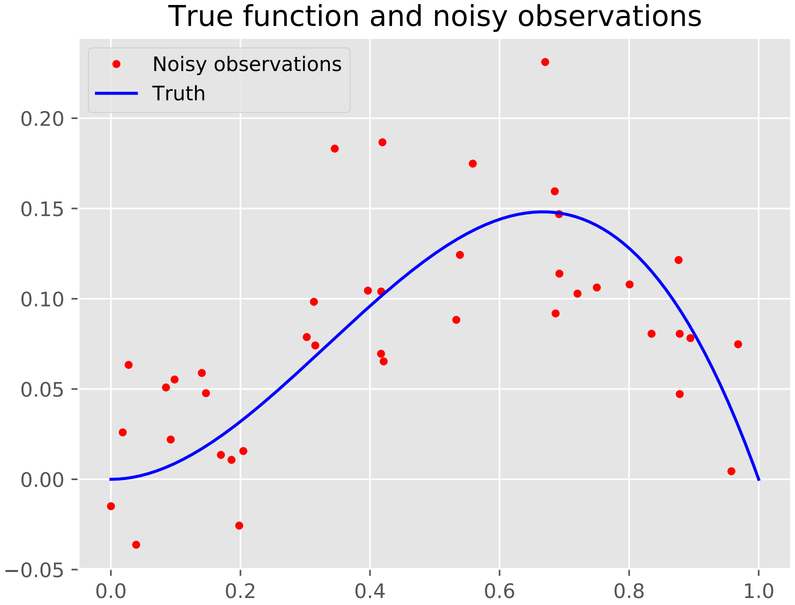
$$f^{t+1} = f^t - \alpha^t g^t \qquad \text{where } g^t = \arg\max_{\|g\|_{\mathscr{F}} \leq 1} \langle g, \, \nabla \mathcal{L}_n(f^t) \rangle$$

[Freund & Schapire, 1997; Mason et al., 1999; Friedman et al., 2000]

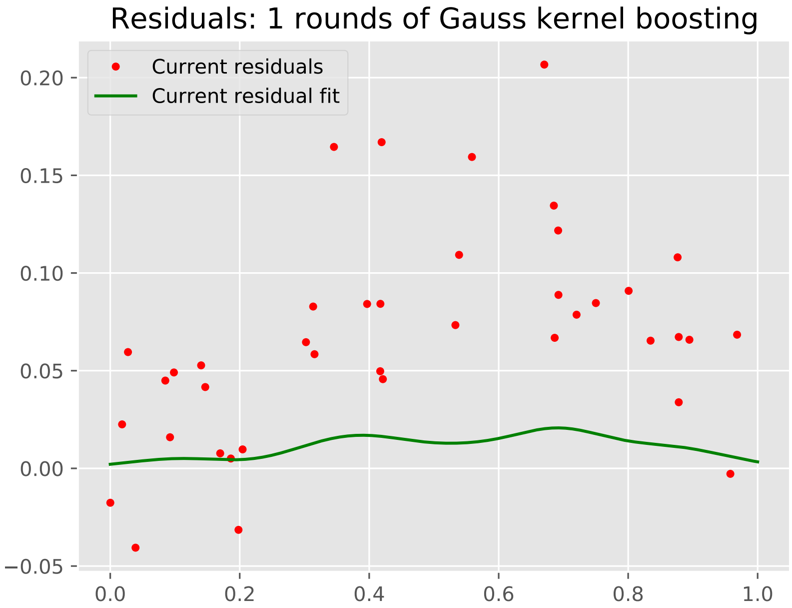
Example: L^2-Boosting with $\mathcal{L}_n(f) = \frac{1}{2n} \sum_{i=1}^n \big[y_i - f(x_i)\big]^2$.
Gradient boosting update takes form

$$g^t = \arg\max_{\|g\|_{\mathscr{F}} \leq 1} \left\{ \frac{1}{n} \sum_{i=1}^n g(x_i) \underbrace{\big[f^t(x_i) - y_i\big]}_{\text{Current residual}} \right\}.$$

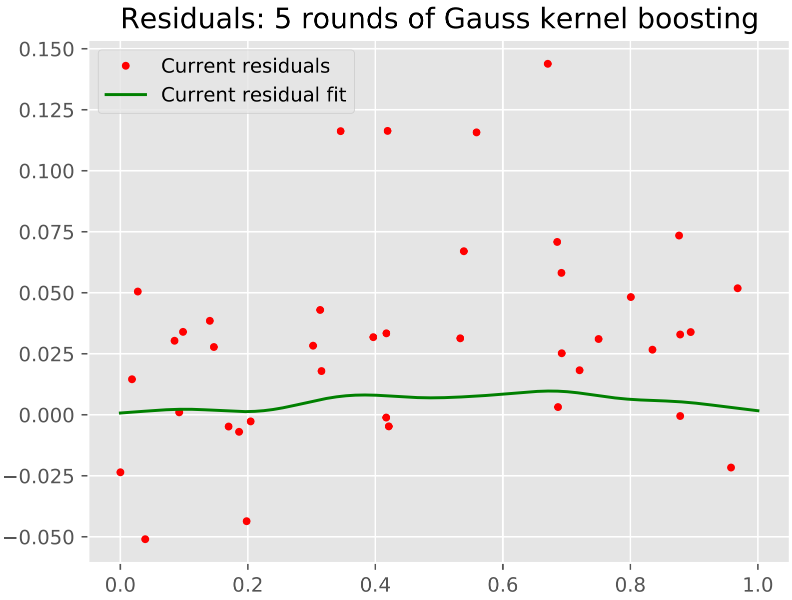
Boosting with a Gaussian kernel



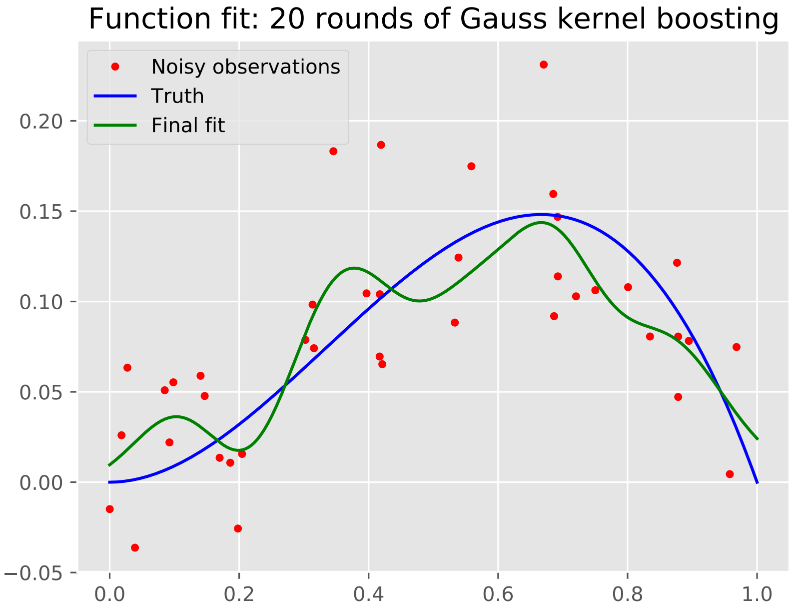
Boosting with a Gaussian kernel



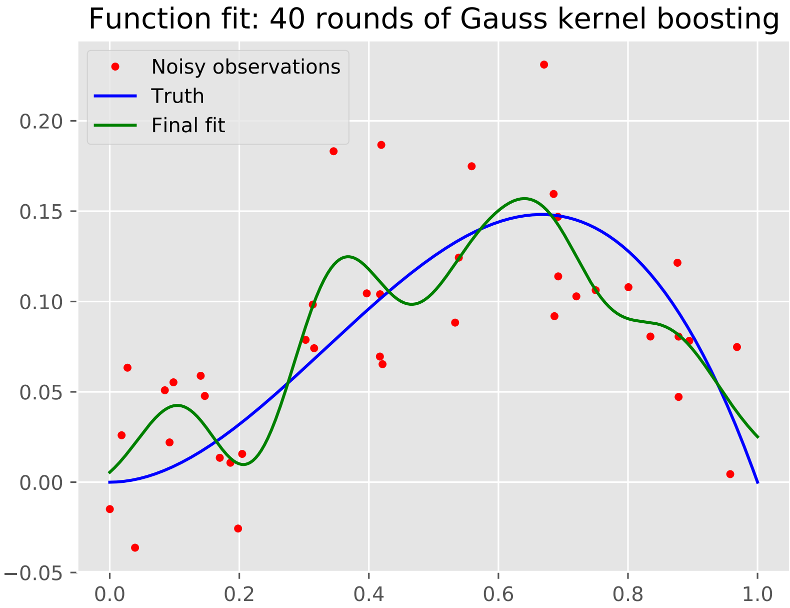
Boosting with a Gaussian kernel



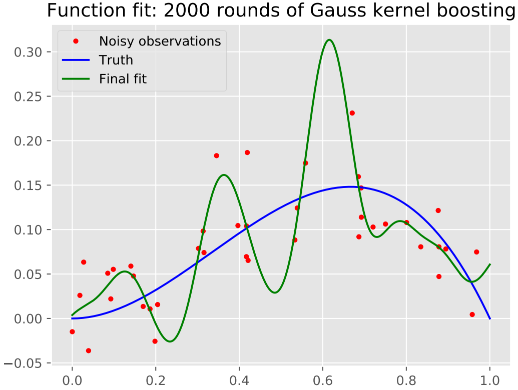
Boosting with a Gaussian kernel



Boosting with a Gaussian kernel



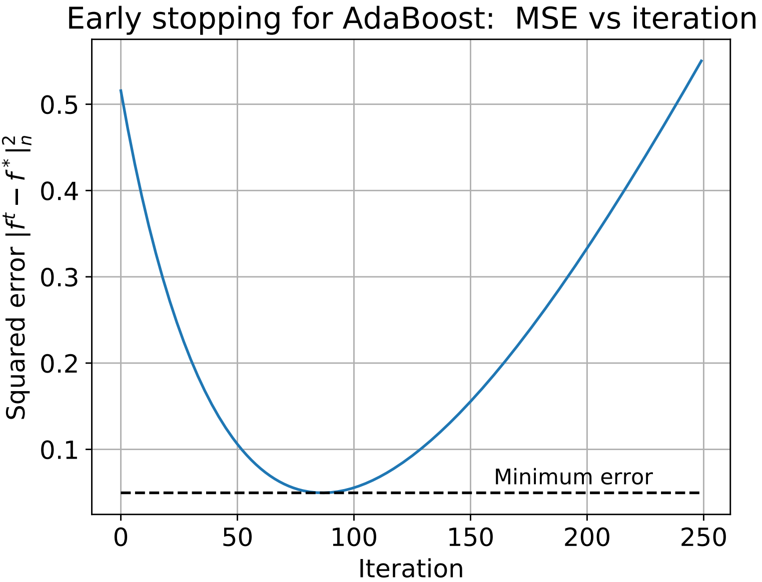
Visualization of over-fitting



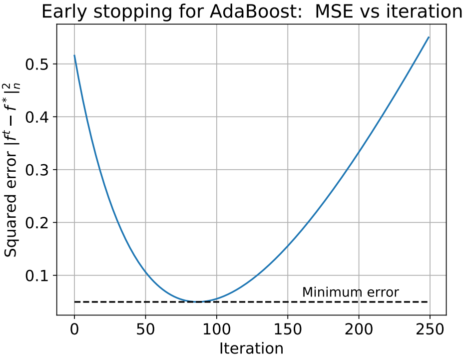
I shall not today attempt further to define the kinds of material I understand to be embraced within that shorthand description "overfitting", and perhaps I could never succeed in intelligibly doing so. But I know it when I see it.

Paraphrased from US Supreme Court Justice Potter Stewart, 1964

How to stop at the “right time”?



How to stop at the "right time"?



Desiderata:

- a data-dependent stopping rule $\{x_i, y_i\}_{i=1}^n \mapsto T \in \{1, 2, \ldots,\}$
- function estimate at iteration t has optimal mean-squared error

$$\|f^T - f^*\|_n^2 \asymp \min_{k=1,2,\ldots} \|f^k - f^*\|_n^2$$

How to stop at the "right time"?

Desiderata:

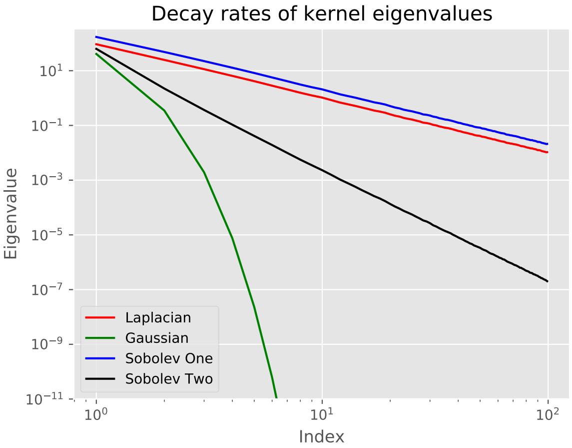
- a data-dependent stopping rule $\{x_i, y_i\}_{i=1}^n \mapsto T \in \{1, 2, \ldots, \}$
- function estimate at iteration t has optimal mean-squared error

$$\|f^T - f^*\|_n^2 \asymp \min_{k=1,2,\ldots} \|f^k - f^*\|_n^2$$

Some past work:

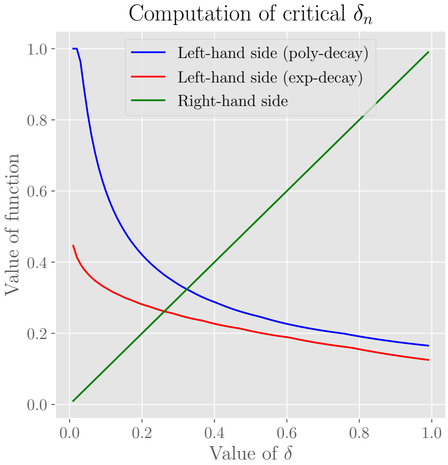
- optimal but not data-dependent rule for spline kernel L^2-boosting: Bühlmann & Yu, 2003
- some results on L^2-boosting with spline kernels: Caponetto et al., 2007, Roscasco et al., 2009
- optimal rates for L^2-boosting, data-dependent any reproducing kernel: Raskutti, W. & Y., 2013

Kernel eigenvalues control "richness"



Kernel	Gaussian	Laplacian	Sobolev One
Form	$\exp(-\frac{1}{2\gamma}(x-y)^2)$	$\exp(-\frac{1}{\gamma}\lvert x-y\rvert)$	$1+\min\{x,y\}$

Statistical error determined by fixed point equation



Fixed point equation:

$$\frac{1}{\sqrt{n}} \sqrt{\sum_{j=1}^{n} \min \left\{ 1, \; \frac{\widehat{\mu}_j^2}{\delta^2} \right\}} = \frac{\delta}{\sigma}$$

where

- $\{\widehat{\mu}_j\}_{j=1}^{n}$ are eigenvalues of kernel matrix
- $\sigma > 0$ is noise level

Required number of iterations scales inversely...

Kernel	Stat. error	Number of iterations
Polynomial (Degree D)	$\frac{D}{n}$	$\frac{n}{D}$
Gaussian	$\frac{\sqrt{\log n}}{n}$	$\frac{n}{\sqrt{\log n}}$
First-order spline	$\left(\frac{1}{n}\right)^{\frac{2}{3}}$	$n^{2/3}$
Second-order spline	$\left(\frac{1}{n}\right)^{\frac{4}{5}}$	$n^{4/5}$

Minimax-optimal bounds for kernel boosting

Kernel boosting sequence:

$$f^{t+1} = f^t - \alpha g^t \qquad \text{where } g^t = \arg \max_{\|g\|_{\mathscr{F}} \leq 1} \langle g, \nabla \mathcal{L}_n(f^t) \rangle$$

Theorem (Wei, Yang & W., 2017)

For any kernel class $\mathscr{F}$, any (m, L)-regular loss function and any step size $\alpha \in (0, \frac{m}{L}]$, and any iterate $t = 1, \ldots, \lfloor \frac{1}{\delta_n^2} \rfloor$:

$$\underbrace{\mathcal{L}(\bar{f}^t) - \mathcal{L}(f^*)}_{\textit{Excess loss}} \quad \precsim \quad \underbrace{\frac{1}{\alpha t}}_{\textit{Opt. error}} \quad + \quad \underbrace{\delta_n^2}_{\textit{Stat. error}}$$

with high probability over the randomized realization.

Minimax-optimal bounds for kernel boosting

Theorem (Wei, Yang & W., 2017)

For any kernel class $\mathscr{F}$, any (m, L)-regular loss function and any step size $\alpha \in (0, \frac{m}{L}]$, and any iterate $t = 1, \ldots, \lfloor \frac{1}{\delta_n^2} \rfloor$:

$$\underbrace{\mathcal{L}(\bar{f}^t) - \mathcal{L}(f^*)}_{\textit{Excess loss}} \quad \precsim \quad \underbrace{\frac{1}{\alpha t}}_{\textit{Opt. error}} \quad + \quad \underbrace{\delta_n^2}_{\textit{Stat. error}}$$

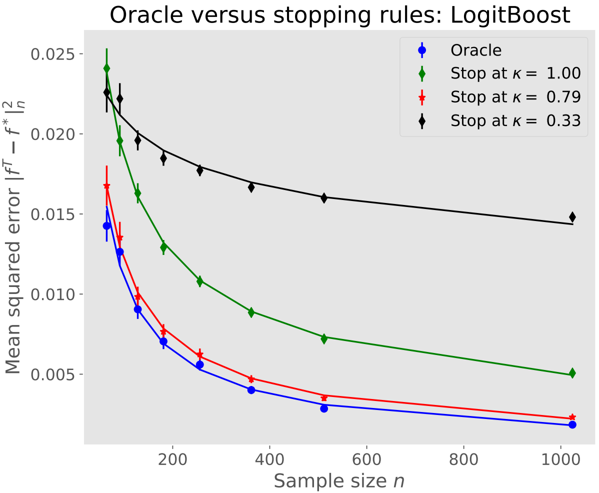
with high probability over the randomized realization.

Statistical error determined by fixed point equation:

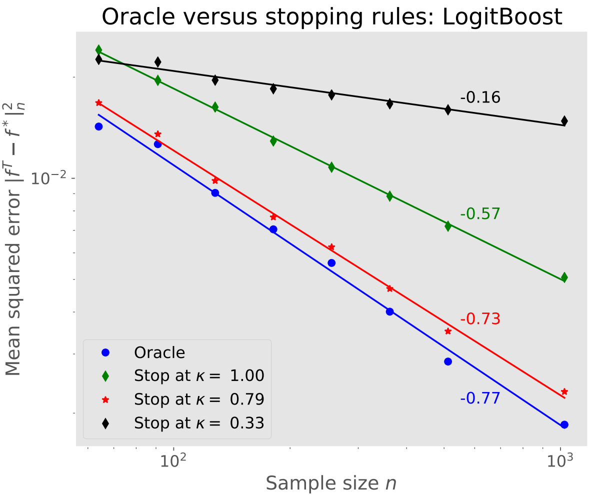
$$\frac{1}{\sqrt{n}} \sqrt{\sum_{j=1}^{n} \min\left\{1, \; \frac{\widehat{\mu}_j^2}{\delta^2}\right\}} = \frac{\delta}{\sigma}$$

where $\{\widehat{\mu}_j\}_{j=1}^n$ are eigenvalues of kernel matrix, and $\sigma > 0$ is noise level.

LogitBoost: Error on linear scale



LogitBoost: Error on logarithmic scale



Optimal theoretical rate: $\left(\frac{\sigma^2}{n}\right)^{0.79}$

Summary

Many challenges lie at the interface of statistics and optimization:

- data sketches for randomized dimension reduction
- regularization via early stopping of iterative algorithms for optimization

<u>Some papers:</u>

Pilanci & W. (2016). Iterative Hessian sketch: Fast and accurate solution approximation for constrained least squares *J. Machine Learning Research.*

Pilanci & W. (2017). Newton sketch: A linear-time optimization algorithm with linear-quadratic convergence. *SIAM Journal on Optimization.*

Wei, Yang & W. (2017). Early stopping for kernel boosting algorithms. Arxiv pre-print.

Fast Johnson-Lindenstrauss sketch

Step 1: Choose some fixed orthonormal matrix $H \in \mathbb{R}^{n \times n}$.
Example: Hadamard matrices

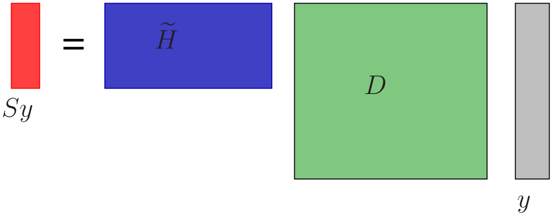
$$H_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \qquad H_{2^t} = \underbrace{H_2 \otimes H_2 \otimes \cdots \otimes H_2}_{\text{Kronecker product } t \text{ times}}$$

(E.g., Ailon & Liberty, 2010)

Fast Johnson-Lindenstrauss sketch

Step 1: Choose some fixed orthonormal matrix $H \in \mathbb{R}^{n \times n}$.
Example: Hadamard matrices

$$H_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \qquad H_{2^t} = \underbrace{H_2 \otimes H_2 \otimes \cdots \otimes H_2}_{\text{Kronecker product } t \text{ times}}$$



Step 2:

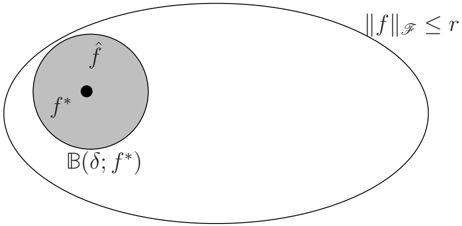
(A) Multiply data vector y with a diagonal matrix of random signs $\{-1, +1\}$

(B) Choose m rows of H to form sub-sampled matrix $\widetilde{H} \in \mathbb{R}^{m \times n}$

(C) Requires $\mathcal{O}(n \log m)$ time to compute sketched vector $Sy = \widetilde{H}\, D y$.

(E.g., Ailon & Liberty, 2010)

Tools for sharp analysis



Localized Gaussian complexity

How much can you align with i.i.d. noise sequence $\{w_i\}_{i=1}^n \sim N(0,1)$?

$$\mathscr{G}_n(\delta, r; \mathscr{F}) = \mathbb{E}_w \sup_{\substack{\|f\|_{\mathscr{F}} \leq r \\ \|f - f^*\| \leq \delta}} \left| \frac{1}{n} \sum_{i=1}^n w_i \big(f(x_i) - f^*(x_i) \big) \right|$$

(e.g., van de Geer, 2000; Bartlett et al., 2005; Koltchinski, 2006)