Fast l_p–regression in a Data Stream

Christian Sohler (TU Dortmund)

David Woodruff (IBM Almaden)

Overview

- Massive data sets
- Streaming algorithms
- Regression
- Clarkson‘s algorithm
- Our results
- Subspace embeddings
- Noisy sampling

Massive data sets

Examples

- Internet traffic logs
- Financial data
- etc.

Streaming algorithms

Scenario

- Data arrives sequentially at a high rate and in arbitrary order
- Data is too large to be stored completely or is stored in secondary memory (where streaming is the fastest way of accessing the data)
- We want some information about the data

Algorithmic requirements

- Data must be processed quickly
- Only a summary of the data can be stored
- Goal: Approximate some statistics of the data

Streaming algorithms

The turnstile model

- Input: A sequence of updates to an object (vector, matrix, database, etc.)
- Output: An approximation of some statistics of the object
- Space: significantly sublinear in input size
- Overall time: near-linear in input size

Streaming algorithms

Example

- Approximating the number of users of a search engine
- Each user has its ID (IP-address)
- Take the vector v of all valid IP-addresses as the object
- Entries of v: #queries submitted to search engine
- Whenever a user submits a query, increment v at the entry corresponding to the submitting IP-address
- Required statistic: # non-zero entries in the current vector

Regression analysis

Regression

- Statistical method to study dependencies between variables in the presence of noise.

Regression analysis

Linear Regression

- Statistical method to study linear dependencies between variables in the presence of noise.

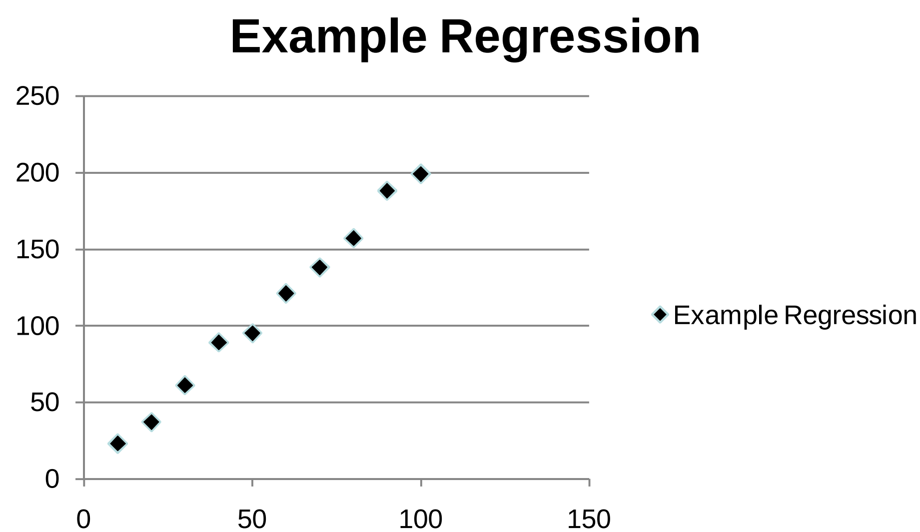
Regression analysis

Linear Regression

- Statistical method to study linear dependencies between variables in the presence of noise.

Example

- Ohm's law V = R · I



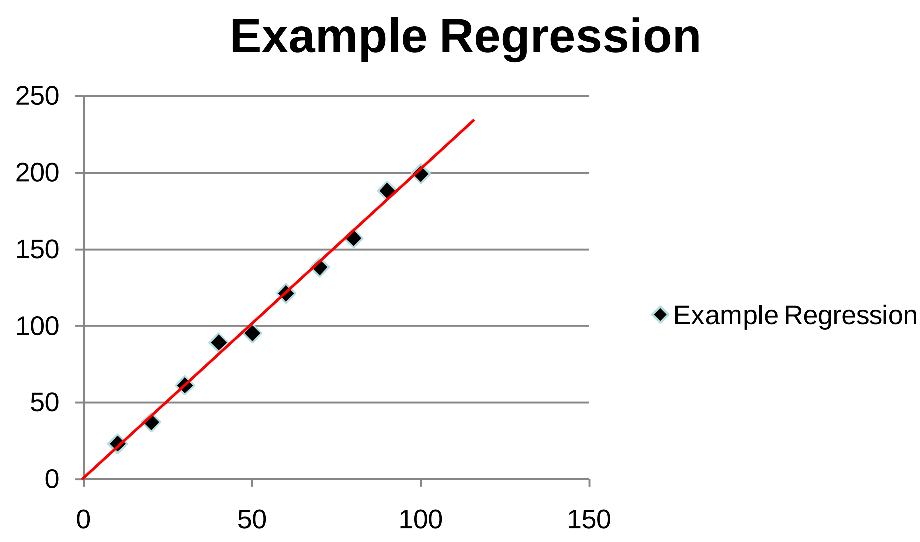
Regression analysis

Linear Regression

- Statistical method to study linear dependencies between variables in the presence of noise.

Example

- Ohm's law $V = R \cdot I$
- Find linear function that best fits the data



Regression analysis

Linear Regression

- Statistical method to study linear dependencies between variables in the presence of noise.

Standard Setting

- One measured variable y
- A set of predictor variables $x_1, \dots, x_d$
- Assumption:

$$y = b_0 + b_1 x_1 + \dots + b_d x_d + e$$

- e is assumed to be a noise (random) variable and the b_j are model parameters

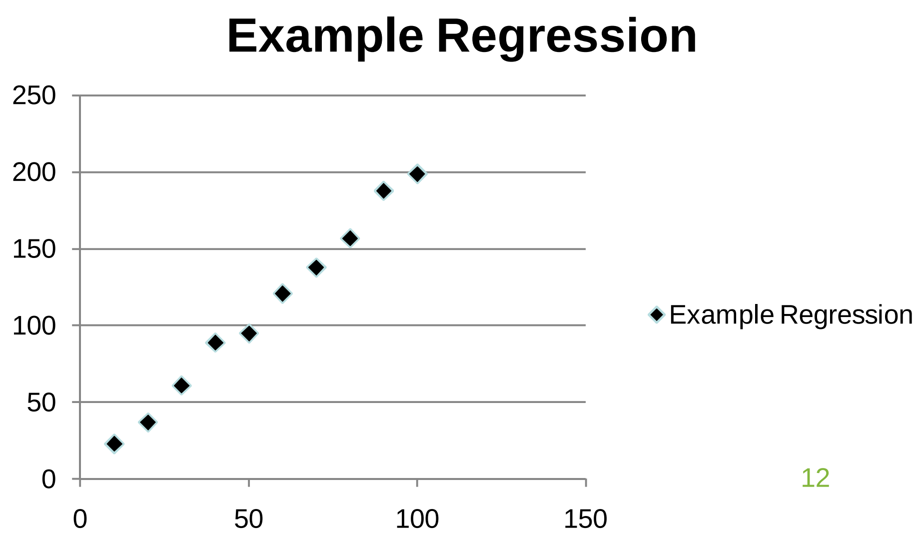
Regression analysis

Example

- Measured variable is the voltage V
- Predictor variable is the current I
- (Unknown) model parameter is the resistance R
- We get pairs of observations for V and I, i.e. pairs (x_i, y_i) where x is some current and y is some measured voltage

Assumption

- Each pair (x,y) was generated through $y = R \cdot x + e$, where e is distributed according to some noise distribution, e.g. Gaussian noise



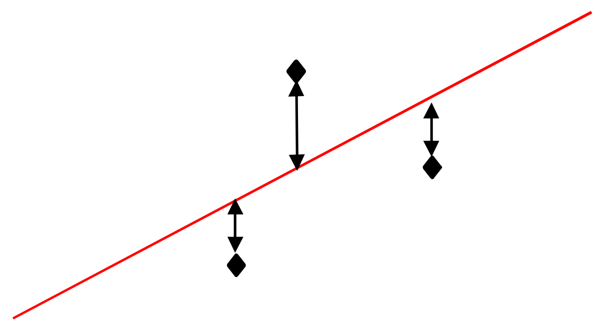
Regression analysis

Setting

- Experimental data is assumed to be generated as pairs (x_i , y_i) with $y_i = b_0 + b_1 x_{i,1} + \ldots + b_d x_{i,d} + e$,
- where e is drawn from some noise distribution, e.g., a Gaussian distribution

Least Squares Method

- Find b* that minimizes $S\,(y_i - b^* x_i)^2$
- Maximizes the (log)-likelihood of b, i.e. the probability density of the y_i given b
- Other desirable statistical properties



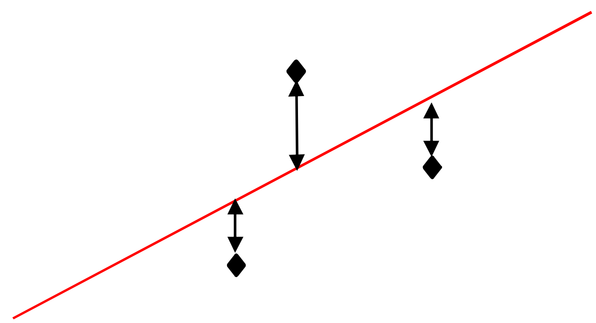
Regression analysis

Model

- Experimental data is assumed to be generated as pairs (x_i , y_i) with $y_i = b_0 + b_1\ x_{i,1} + \ldots + b_d\ \ x_{i,d}\ + e$,
- where e is drawn from some noise distribution, e.g. a Gaussian distribution

Method of least absolute deviation

- Find b* that minimizes $S\ |y_i - b^*\ x_i\ |$
- More robust than least squares



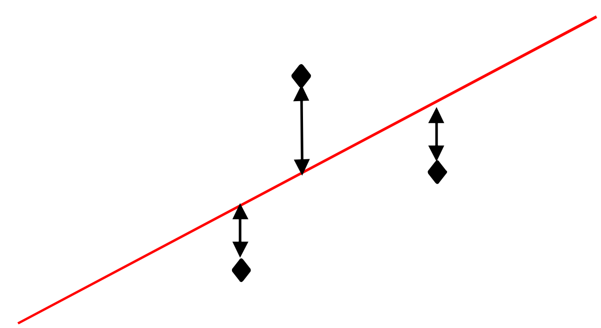
Regression analysis

Model

- Experimental data is assumed to be generated as pairs (x_i, y_i) with $y_i = b_0 + b_1 x_{i,1} + \ldots + b_d x_{i,d} + e$,
- where e is drawn from some noise distribution, e.g. a Gaussian distribution

Method of least absolute deviation (l_1-regression)

- Find b^* that minimizes S $|y_i - b^* x_i|$
- More robust than least squares



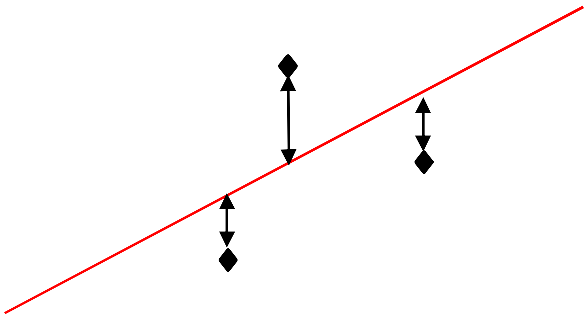
Regression analysis

Model

- Experimental data is assumed to be generated as pairs (x_i, y_i) with
  $y_i = b_0 + b_1 x_{i,1} + \dots + b_d x_{i,d} + e$,
- where e is drawn from some noise distribution, e.g., a Gaussian distribution

l_p-regression

- Find b* that minimizes $\Sigma |y_i - b^* x_i|^p$, $1 < p < 2$
- More robust than least squares



Regression analysis

Matrix form for l_p -regression, $1 \le p \le 2$

- **Input:** $n \times d$-matrix X whose rows are the x_i and a vector $y=(y_1, \ldots, y_n)$
  n is the number of observations; d is the number of predictor variables (We assume that $b_0 = 0$ for all i)

- **Output:** b^* that minimizes $\|Xb^*-y\|_p^p$

Regression analysis

Geometry of regression

- Assume n À d
- We want to find a b* that minimizes $||Xb^*-y||_p^p$
- The product Xb* can be written as

$$X_{*1} b^*_1 + X_{*2} b^*_2 + \ldots + X_{*d} b^*_d$$

- where X_{*i} is the i-th column of X
- This is a linear k-dimensional subspace (k≤d is the rank of X)
- The problem is equivalent to computing the point of the column space of X nearest to y in l_p-norm

Regression analysis

$(1+\epsilon)$-approximation algorithm for l_1 - regression [Clarkson, SODA'05]

Input: $n \times d$ matrix X, vector y

Output: vector b' s.t. $||Xb' - y||_1 \leq (1+\epsilon) \cdot ||Xb^* - y||_1$

1. Compute O(1)-Approximation b''
2. Compute the residual r' = Xb''-y
3. Scale r' such that $||r'||_1 = d$
4. Compute a well-conditioned basis U of the column space of X
5. Sample row i according to $p_i = f_i \cdot \text{poly}(d, 1/\epsilon)$ where $f_i = |r'_i| + ||u_{i*}|| / (|r'| + ||U||)$
6. Assign to each sample row a weight of $1/p_i$
7. Solve the problem on the sample set using linear programming

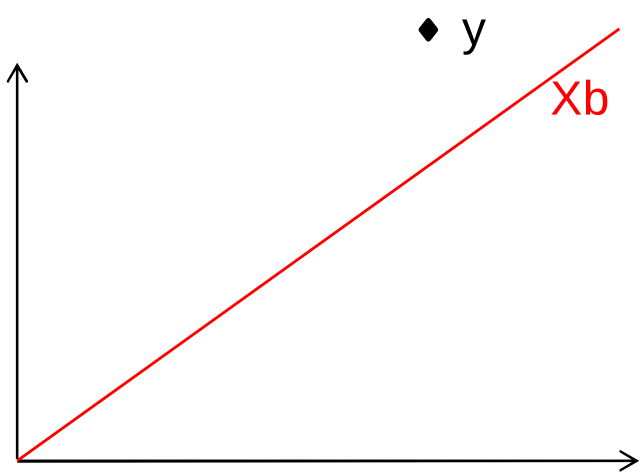
Regression analysis

(1+e)-approximation algorithm for l_1 - regression [Clarkson, SODA'05]

Input: $n \times d$ matrix X, vector y

Output: vector b' s.t. $||Xb' - y||_1 \leq (1+e) \cdot ||Xb^* - y||_1$

1. Compute O(1)-Approximation b''
2. Compute the residual r' = Xb''-y
3. Scale r' such that $||r'||_1 = d$
4. Compute a well-conditioned basis U of the column space of X
5. Sample row i according to $p_i = f_i \cdot \text{poly}(d, 1/e)$ where $f_i = |r'_i| + ||u_{i*}|| / (|r'| + ||U||)$
6. Assign to each sample row a weight of $1/p_i$
7. Solve the problem on the sample set using linear programming



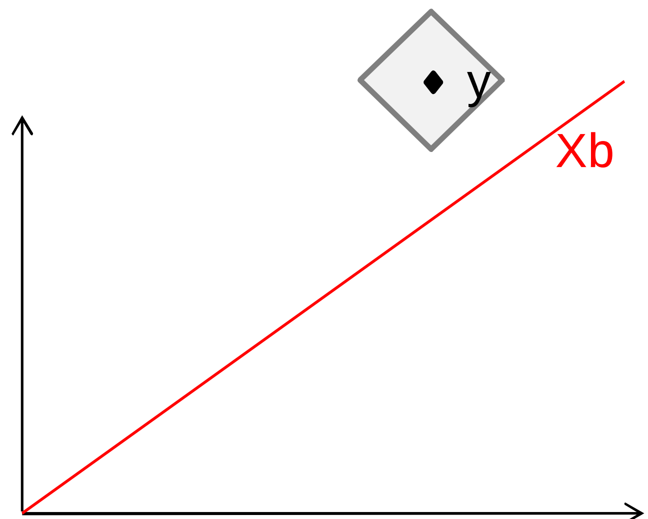
Regression analysis

(1+e)-approximation algorithm for l_1 - regression [Clarkson, SODA'05]

Input: $n \times d$ matrix X, vector y

Output: vector b' s.t. $||Xb' - y||_1 \leq (1+e) \cdot ||Xb^* - y||_1$

1. Compute O(1)-Approximation b''
2. Compute the residual r' = Xb''-y
3. Scale r' such that $||r'||_1 = d$
4. Compute a well-conditioned basis U of the column space of X
5. Sample row i according to $p_i = f_i \cdot \mathrm{poly}(d, 1/e)$ where $f_i = |r'_i| + ||u_{i*}|| / (|r'| + ||U||)$
6. Assign to each sample row a weight of $1/p_i$
7. Solve the problem on the sample set using linear programming



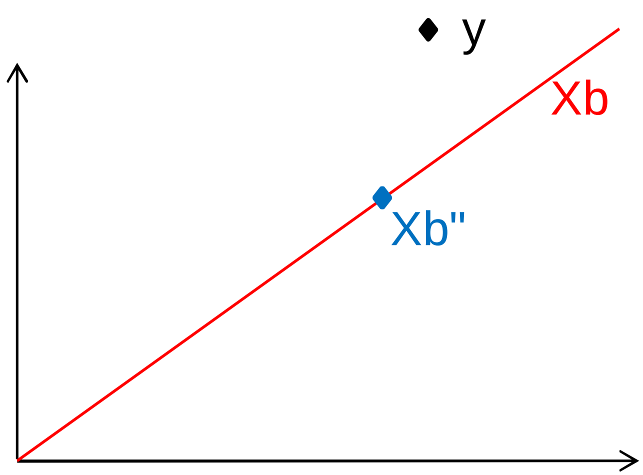
Regression analysis

(1+e)-approximation algorithm for l_1 - regression [Clarkson, SODA'05]

Input: $n \times d$ matrix X, vector y

Output: vector b' s.t. $||Xb' - y||_1 \leq (1+e) \cdot ||Xb^* - y||_1$

1. Compute O(1)-Approximation b''
2. Compute the residual r' = Xb''-y
3. Scale r' such that $||r'||_1 = d$
4. Compute a well-conditioned basis U of the column space of X
5. Sample row i according to $p_i = f_i \cdot \text{poly}(d, 1/e)$ where $f_i = |r'_i| + ||u_{i*}|| / (|r'| + ||U||)$
6. Assign to each sample row a weight of $1/p_i$
7. Solve the problem on the sample set using linear programming

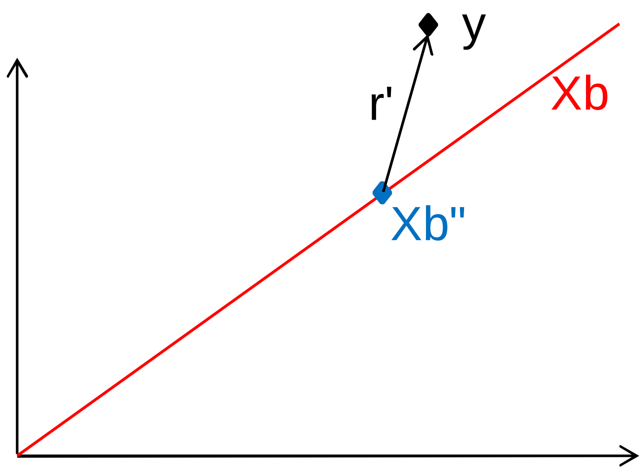


Regression analysis

(1+e)-approximation algorithm for l_1 - regression [Clarkson, SODA'05]

Input: $n \times d$ matrix X, vector y

Output: vector b' s.t. $||Xb' - y||_1 \leq (1+e) \cdot ||Xb^* - y||_1$



1. Compute O(1)-Approximation b"
2. Compute the residual r' = Xb"-y
3. Scale r' such that $||r'||_1 = d$
4. Compute a well-conditioned basis U of the column space of X
5. Sample row i according to $p_i = f_i \cdot \mathrm{poly}(d, 1/e)$ where $f_i = |r'_i| + ||u_{i*}|| / (|r'| + ||U||)$
6. Assign to each sample row a weight of $1/p_i$
7. Solve the problem on the sample set using linear programming

Regression analysis

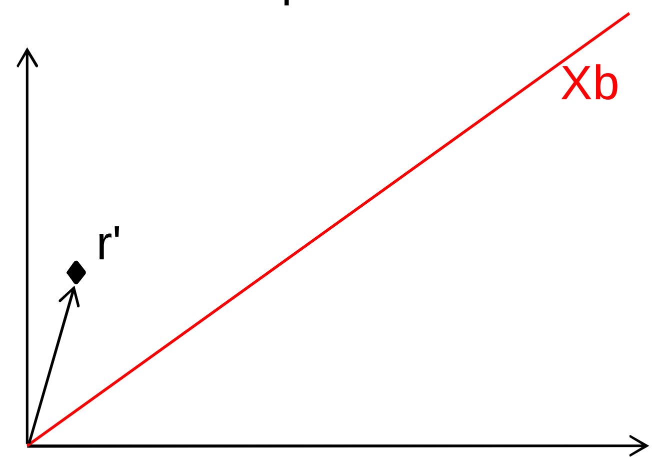
(1+e)-approximation algorithm for l_1 - regression [Clarkson, SODA'05]

Input: $n \times d$ matrix X, vector y

Output: vector b' s.t. $||Xb' - y||_1 \leq (1+e) \cdot ||Xb^* - y||_1$

1. Compute O(1)-Approximation b''
2. Compute the residual r' = Xb''-y
3. Scale r' such that $||r'||_1 = d$
4. Compute a well-conditioned basis U of the column space of X
5. Sample row i according to $p_i = f_i \cdot \text{poly}(d, 1/e)$ where $f_i = |r'_i| + ||u_{i*}|| / (|r'| + ||U||)$
6. Assign to each sample row a weight of $1/p_i$
7. Solve the problem on the sample set using linear programming

New problem:



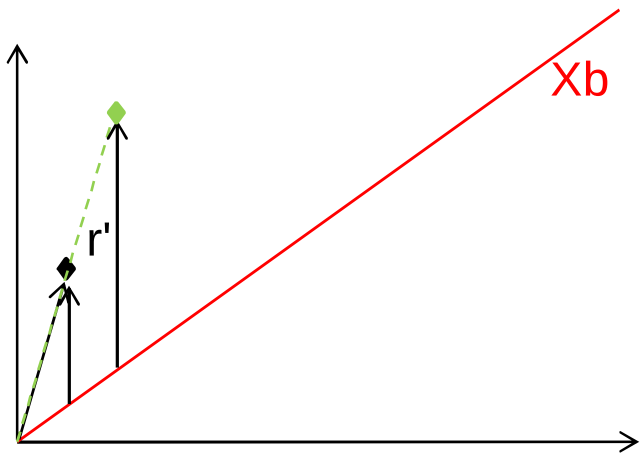
Regression analysis

(1+e)-approximation algorithm for l_1 - regression [Clarkson, SODA'05]

Input: $n \times d$ matrix X, vector y

Output: vector b' s.t. $||Xb' - y||_1 \leq (1+e) \cdot ||Xb^* - y||_1$

1. Compute O(1)-Approximation b''
2. Compute the residual r' = Xb''-y
3. $\boxed{\text{Scale } r' \text{ such that } ||r'||_1 = d}$
4. Compute a well-conditioned basis U of the column space of X
5. Sample row i according to $p_i = f_i \cdot \text{poly}(d, 1/e)$ where $f_i = |r'_i| + ||u_{i*}|| / (|r'| + ||U||)$
6. Assign to each sample row a weight of $1/p_i$
7. Solve the problem on the sample set using linear programming



Regression analysis

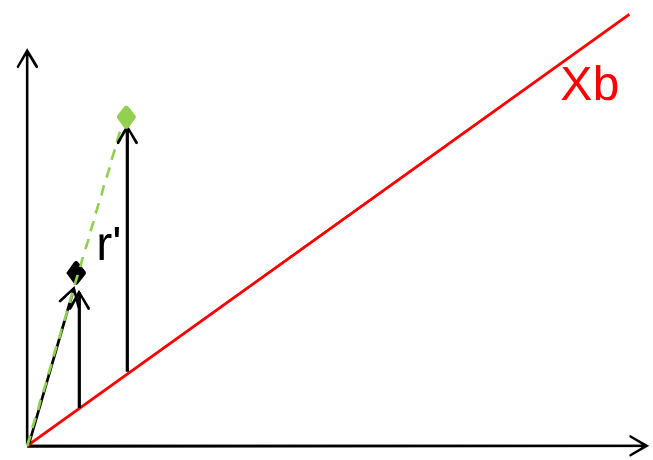
(1+e)-approximation algorithm for l_1 - regression [Clarkson, SODA'05]

Input: $n \times d$ matrix X, vector y

Output: vector b' s.t. $||Xb' - y||_1 \leq (1+e) \cdot ||Xb^* - y||_1$

Well-conditioned basis U:
Basis with
$||z||_1 \cdot ||Uz||_1 \cdot \text{poly}(d)||z||_1$

1. Compute O(1)-Approximation b''
2. Compute the residual r' = Xb''-y
3. Scale r' such that $||r'||_1 = d$
4. Compute a well-conditioned basis U of the column space of X
5. Sample row i according to $p_i = f_i \cdot \text{poly}(d, 1/e)$ where $f_i = |r'_i| + ||u_{i*}|| / (|r'| + ||U||)$
6. Assign to each sample row a weight of $1/p_i$
7. Solve the problem on the sample set using linear programming



Regression analysis

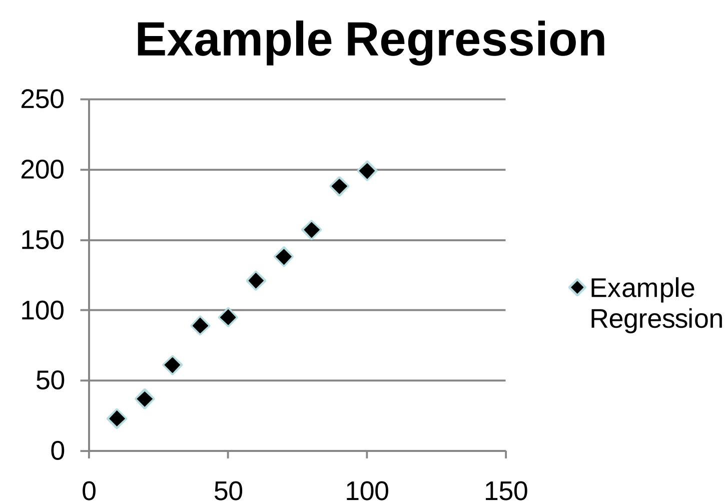
(1+e)-approximation algorithm for l_1 - regression [Clarkson, SODA'05]

Input: $n \times d$ matrix X, vector y

Output: vector b' s.t. $||Xb' - y||_1 \leq (1+e) \cdot ||Xb^* - y||_1$

1. Compute O(1)-Approximation b''
2. Compute the residual r' = Xb''-y
3. Scale r' such that $||r'||_1 = d$
4. Compute a well-conditioned basis U of the column space of X
5. Sample row i according to $p_i = f_i \cdot \text{poly}(d, 1/e)$ where $f_i = |r'_i| + ||u_{i*}|| / (|r'| + ||U||)$
6. Assign to each sample row a weight of $1/p_i$
7. Solve the problem on the sample set using linear programming

Well-conditioned basis U:
Basis with
$||z||_1 \cdot ||Uz||_1 \cdot \text{poly}(d)||z||_1$



Regression analysis

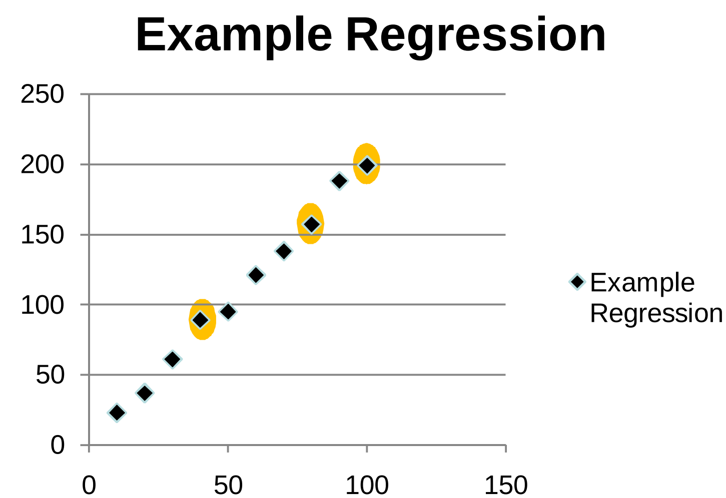
(1+e)-approximation algorithm for l_1 - regression [Clarkson, SODA'05]

Input: $n \times d$ matrix X, vector y

Output: vector b' s.t. $||Xb' - y||_1 \leq (1+e) \cdot ||Xb^* - y||_1$

1. Compute O(1)-Approximation b''
2. Compute the residual r' = Xb''-y
3. Scale r' such that $||r'||_1 = d$
4. Compute a well-conditioned basis U of the column space of X
5. Sample row i according to $p_i = f_i \cdot poly(d, 1/e)$ where $f_i = |r'_i| + ||u_{i*}|| / (|r'| + ||U||)$
6. Assign to each sample row a weight of $1/p_i$
7. Solve the problem on the sample set using linear programming

Well-conditioned basis U:
Basis with
$||z||_1 \cdot ||Uz||_1 \cdot poly(d)||z||_1$



Regression analysis

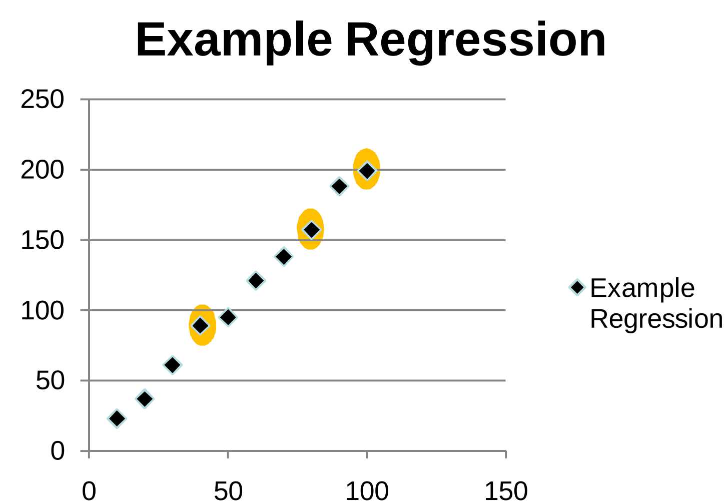
(1+e)-approximation algorithm for l_1 - regression [Clarkson, SODA'05]

Input: $n \times d$ matrix X, vector y

Output: vector b' s.t. $||Xb' - y||_1 \leq (1+e) \cdot ||Xb^* - y||_1$

1. Compute O(1)-Approximation b''
2. Compute the residual r' = Xb''-y
3. Scale r' such that $||r'||_1 = d$
4. Compute a well-conditioned basis U of the column space of X
5. Sample row i according to $p_i = f_i \cdot \text{poly}(d, 1/e)$ where $f_i = |r'_i| + ||u_{i*}|| / (|r'| + ||U||)$
6. Assign to each sample row a weight of $1/p_i$
7. Solve the problem on the sample set using linear programming

Well-conditioned basis U:
Basis with
$||z||_1 \cdot ||Uz||_1 \cdot \text{poly}(d)||z||_1$



Regression analysis

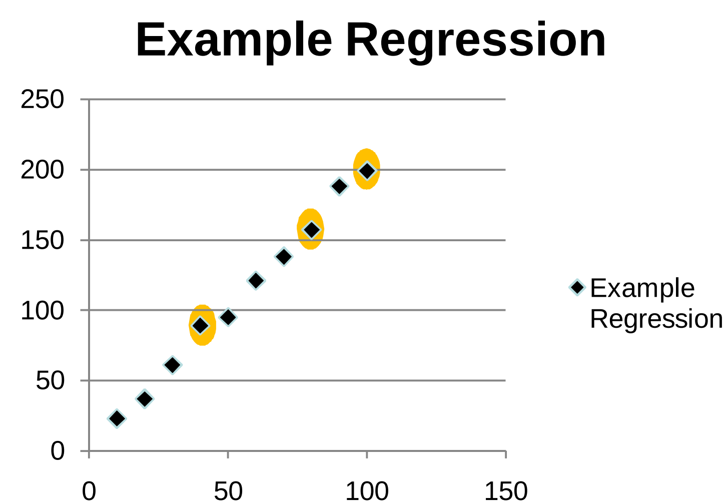
(1+e)-approximation algorithm for l_1 - regression [Clarkson, SODA'05]

Input: $n \times d$ matrix X, vector y

Output: vector b' s.t. $||Xb' - y||_1 \leq (1+e) \cdot ||Xb^* - y||_1$

1. Compute O(1)-Approximation b''
2. Compute the residual r' = Xb''-y
3. Scale r' such that $||r'||_1 = d$
4. Compute a well-conditioned basis U of the column space of X
5. Sample row i according to $p_i = f_i \cdot \text{poly}(d, 1/e)$ where $f_i = |r'_i| + ||u_{i*}|| / (|r'| + ||U||)$
6. Assign to each sample row a weight of $1/p_i$
7. Solve the problem on the sample set using linear programming

Well-conditioned basis U:
Basis with
$||z||_1 \cdot ||Uz||_1 \cdot \text{poly}(d)||z||_1$



Regression analysis

Solving l_1-regression via linear programming

- Minimize $(1,\ldots,1) \cdot (a^{+} + a^{-})$
- Subject to:

$$X b + a^{+} - a^{-} = y$$
$$a^{+}, a^{-} \geq 0$$

Regression for data streams

l_1-regression

- X: n×d-matrix of predictor variables, n is the number of observations
- y: vector of measured variables
- b: unknown model parameter (this is what we want to optimize)
- Find b that minimizes $||Xb-y||_1$

Turnstile model

- We get updates for X and y
- Example: (i,j,c) means X[i,j] = X[i,j] + c
- Heavily overconstrained case: $n \gg d$

Regression for data streams

State of the art

- Small space streaming algorithm in the turnstile model for l_p-regression for all p, $1 \leq p \leq 2$; the time to extract the solution is prohibitively large [Feldman, Monemizadeh, Sohler, W; SODA'10]
- Efficient streaming algorithm in the turnstile model for l_2-regression [Clarkson, W, STOC'09]
- Somewhat efficient non-streaming (1+e)-approximations for l_p–regression [Clarkson, SODA'05; Drineas, Mahoney, Muthukrishnan; SODA'06; Sarlos; FOCS'06; Dasgupta, Drineas, Harb, Kumar, Mahoney; SICOMP'09]

Our Results

- A (1+ε)-approximation algorithm for l_p-regression problem for any p in [1, 2]
  - First 1-pass algorithm in the turnstile model
  - Space complexity poly(d log n / ε)
  - Time complexity $nd^{1.376}$ poly(log n / ε)
  - Improves earlier nd^5 log n time algorithms for every p

- New linear oblivious embeddings from l_p^n to l_p^r
  - r = poly(d log n)
  - Preserve d-dimensional subspaces
  - Distortion is poly(d)

- This talk will focus on the case p = 1

Regression for data streams

First approach

- Leverage Clarkson's algorithm

Sequential structure is hard to implement in streaming

Compute O(1)-approximation

Compute well-conditioned basis

Sample rows from the well-conditioned basis and the residual

Regression for data streams

Theorem 1(l_1 -subspace embedding)

- Let $r \geq poly(d, \ln n)$. There is a probability space over $r \times n$ matrices R such that for any $n \times d$-matrix A with probability at least 99/100 we have for all $b \in \mathbb{R}^d$:

$$||Ab||_1 \leq ||RAb||_1 \leq O(d^2) \cdot ||Ab||_1$$

- R is a scaled matrix of i.i.d. Cauchy random variables
- Argues through the existence of well-conditioned bases
  - Uses "well-conditioned nets"
- Generalizes to p > 1

Regression for data streams

The algorithm – part 1

- Pick random matrix R according to the distribution from the previous theorem
- Maintain RX and Ry during the stream
- Find b' that minimizes ||RXb'-Ry|| using linear programming
- Compute a well-conditioned basis U for RX
- Compute Y such that U = RXY

Lemma 2

With probability 99/100, XY is a well-conditioned basis for the column space of X.

Regression for data streams

R can be stored implicitly.

The algorithm – part 1

- Pick random matrix R according to the distribution from the previous theorem
- Maintain RX and Ry during the streaming
- Find b' that minimizes ||RXb'-Ry|| using linear programming
- Compute a well-conditioned basis U for RX
- Compute Y such that U = RXY

Lemma 2

With probability 99/100, XY is a well-conditioned basis for the column space of X.

Regression for data streams

The algorithm – part 1

- Pick random matrix R according to the distribution from the previous theorem
- Maintain RX and Ry during the streaming
- Find b' that minimizes ||RXb'-Ry|| using linear programming
- Compute a well-conditioned basis U for RX
- Compute Y such that U = RXY

R(X+D) = RX + RD

Lemma 2

With probability 99/100, XY is a well-conditioned basis for the column space of X.

Regression for data streams

The algorithm – part 1

- Pick random matrix R according to the distribution from the previous theorem
- Maintain RX and Ry during the streaming
- Find b' that minimizes ||RXb'-Ry|| using linear programming
- Compute a well-conditioned basis U for RX
- Compute Y such that U = RXY

Lemma 2

With probability 99/100, XY is a well-conditioned basis for the column space of X.

Regression for data streams

The algorithm – part 1

- Pick random matrix R according to the distribution from the previous theorem
- Maintain RX and Ry during the streaming
- Find b' that minimizes ||RXb'-Ry|| using linear programming
- Compute a well-conditioned basis U for RX
- Compute Y such that U = RXY

Using [Clarkson; SODA'05] or [Dasgutpta et. al.; SICOMP09]

Lemma 2

With probability 99/100, XY is a well-conditioned basis for the column space of X.

Regression for data streams

The algorithm – part 1

- Pick random matrix R according to the distribution from the previous theorem
- Maintain RX and Ry during the streaming
- Find b' that minimizes ||RXb'-Ry|| using linear programming
- Compute a well-conditioned basis U for RX
- Compute Y such that U = RXY

The span of U equals the span of RX

Lemma 2

With probability 99/100, XY is a well-conditioned basis for the column space of X.

Regression for data streams

The algorithm – part 1

- Pick random matrix R according to the distribution from the previous theorem
- Maintain RX and Ry during the streaming
- Find b' that minimizes ||RXb'-Ry|| using linear programming
- Compute a well-conditioned basis U for RX
- Compute Y such that U = RXY

Lemma 2

With probability 99/100, XY is a well-conditioned basis for the column space of X.

Regression for data streams

Intermediate summary

- Can compute poly(d)-approximation
- Can compute Y s.t. XY is well-conditioned

Compute O(1)-approximation

Compute well-conditioned basis

Sample rows from the well-conditioned basis and the residual

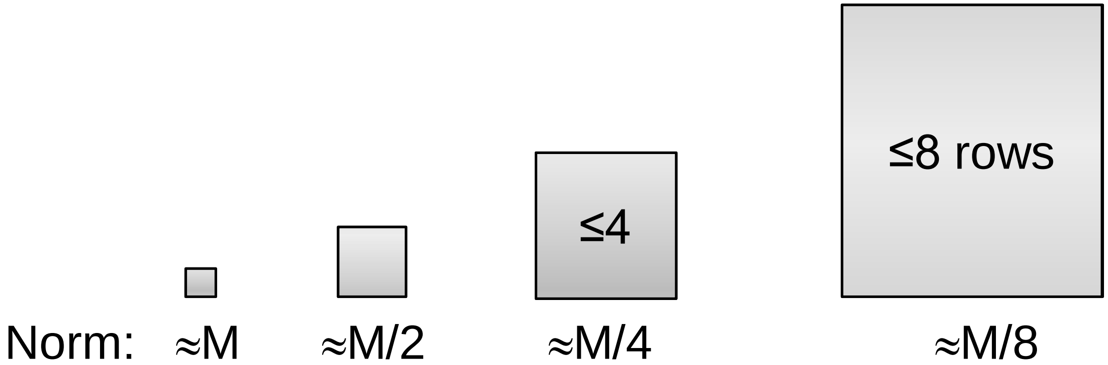
Regression for data streams

We can reduce everything to a new problem

- Updates to matrix B
- Need to sample rows from B with probability according to their l_1-norm
- Assume we know $M=||B||_1$

Noisy sampling [Extension of Andoni, DoBa, Indyk, W; FOCS'09]

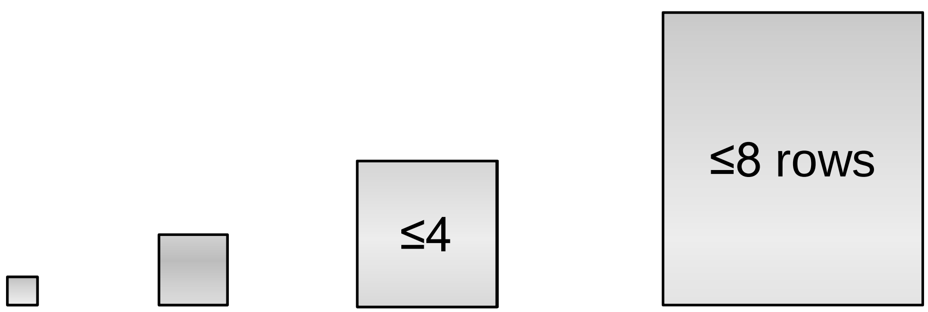
- Subdivide rows into groups



Regression for data streams

Noisy sampling

- Subdivide rows into groups
- Try to sample from each group separately



Norm:	≈M	≈M/2	≈M/4	≈M/8
Prob.:	1	1/2	1/4	1/8

Regression for data streams

Noisy sampling

- Subdivide rows into groups
- Try to sample from each group separately
- **Problem:** Can't store the sample in the stream

≤8 rows

Norm: ≈M/8

Prob.: 1/8

Regression for data streams

Noisy sampling

- Subdivide rows into groups
- Try to sample from each group separately
- **Problem:** Can't store the sample
- **Instead:** Subsampling

≤8 rows

Norm: ≈M/8

Prob.: 1/8

Regression for data streams

Noisy Sampling

- **Grouping:**
- $I_j = \{i : \|B_i\|_1 \in (M/2^j, 2M/2^j]\}$

- **Sample step (Group I_j):**
- Subsample rows with probability $1/2^j$
- Hash sampled rows into w buckets
- Maintain sum of each bucket
- Noise in a bucket ¼ $M/(2^j w)$

- **Verification step:**
- Check if bucket has norm approx. $M/2^j$
- If yes, then return bucket as noisy sample with weight 2^j

Regression for data streams

Summary of the algorithm

- Maintain RX and Ry to obtain poly(d)-approximation and access to matrix B
- Sample rows using our noisy sampling data structure
- Solve the problem on the noisy sample

Regression for data streams

Some simplifications

- Let B be the matrix XY adjunct r' = Xb'-y
- Assume the stream has updates for B

Regression for data streams

Some simplifications

- Let B be the matrix XY adjunct r' = Xb'-y
- Assume the stream has updates for B

Assume we know Y in advance:
(X+D)Y = XY+ DY

Why don‘t we need another pass for this?

- We can treat the entries of Y and b' as formal variables and plug in the values at the end of the stream

Theorem

The above algorithm is a $(1+e)$-approximation to the l_1-regression problem

- uses poly(d, log n, 1/e) space
- implementable in 1-pass in the turnstile model
- can be implemented in $nd^{1.376}$ poly(log n / ε) time
  - Main point is that well-conditioned basis computed in sketch-space

Conclusion

Main results

- First efficient streaming algorithm for l_p-regression, $1 \leq p < 2$
- $nd^{1.376}$ running time improves previous nd^5 running time
- First oblivious poly(d) subspace embedding for l_1

Open problems

- Streaming and/or approximation algorithms for even more robust regression problems like least median of squares, etc.
- Regression when $d \gg n$ (redundant parameters, structural restrictions, …)
- Kernel methods
- Algorithms for statistical problems on massive data sets
- Other applications of our subspace embedding