Fast Algorithms for Segmented Regression

Jayadev Acharya[1] Ilias Diakonikolas[2] Jerry Li[1] Ludwig Schmidt[1]

[1]MIT [2]USC

June 21, 2016

Statistical vs computational tradeoffs?

General Motivating Question

When is it worth it to trade *statistical efficiency* for *runtime*?

Statistical vs computational tradeoffs?

General Motivating Question

When is it worth it to trade *statistical efficiency* for *runtime*?

Given two estimators:

Statistical vs computational tradeoffs?

General Motivating Question

When is it worth it to trade *statistical efficiency* for *runtime*?

Given two estimators:

Estimator A: Great statistical rate, but slow to compute

Estimator B: Worse statistical rate, but faster to compute

Statistical vs computational tradeoffs?

General Motivating Question

When is it worth it to trade *statistical efficiency* for *runtime*?

Given two estimators:

Estimator A: Great statistical rate, but slow to compute

Estimator B: Worse statistical rate, but faster to compute

When is it better to use estimator B vs estimator A?

Statistical vs computational tradeoffs?

General Motivating Question

When is it worth it to trade *statistical efficiency* for *runtime*?

Given two estimators:

Estimator A: Great statistical rate, but slow to compute

Estimator B: Worse statistical rate, but faster to compute

When is it better to use estimator B vs estimator A?

"As data grows, it may be beneficial to consider faster inferential algorithms, because the increasing statistical strength of the data can compensate for the poor algorithmic quality." [Jor13]

Outline

- Introduction
- The exact algorithm
- Our algorithm
- Experiments

Outline

- **Introduction**
- The exact algorithm
- Our algorithm
- Experiments

Linear regression

Linear regression

We are given a labelled data set $(\boldsymbol{x}^{(1)}, y^{(1)}), \ldots, (\boldsymbol{x}^{(n)}, y^{(n)}) \in \mathbb{R}^d \times \mathbb{R}$ so that

$$y^{(i)} = \ell^*(\boldsymbol{x}^{(i)}) + \epsilon^{(i)} \ ,$$

Linear regression

We are given a labelled data set $(\boldsymbol{x}^{(1)}, y^{(1)}), \ldots, (\boldsymbol{x}^{(n)}, y^{(n)}) \in \mathbb{R}^d \times \mathbb{R}$ so that

$$y^{(i)} = \ell^*(\boldsymbol{x}^{(i)}) + \epsilon^{(i)} \ ,$$

$\ell^*(\boldsymbol{x}) = \langle \theta^*, \boldsymbol{x} \rangle$ is an unknown linear function that we want to recover.

Linear regression

We are given a labelled data set $(\boldsymbol{x}^{(1)}, y^{(1)}), \dots, (\boldsymbol{x}^{(n)}, y^{(n)}) \in \mathbb{R}^d \times \mathbb{R}$ so that

$$y^{(i)} = \ell^*(\boldsymbol{x}^{(i)}) + \epsilon^{(i)} \ ,$$

$\ell^*(\boldsymbol{x}) = \langle \theta^*, \boldsymbol{x} \rangle$ is an unknown linear function that we want to recover.

Assume that $\epsilon^{(i)}$ are independent noise variables.

Linear regression

We are given a labelled data set $(\boldsymbol{x}^{(1)}, y^{(1)}), \ldots, (\boldsymbol{x}^{(n)}, y^{(n)}) \in \mathbb{R}^d \times \mathbb{R}$ so that

$$y^{(i)} = \ell^*(\boldsymbol{x}^{(i)}) + \epsilon^{(i)} \ ,$$

$\ell^*(\boldsymbol{x}) = \langle \theta^*, \boldsymbol{x} \rangle$ is an unknown linear function that we want to recover.

Assume that $\epsilon^{(i)}$ are independent noise variables.

Goal: Find a linear $\ell(\boldsymbol{x})$ minimizing

$$\mathrm{MSE}(\ell) = \frac{1}{n} \sum_{i=1}^{n} (\ell(\boldsymbol{x}^{(i)}) - \ell^*(\boldsymbol{x}^{(i)}))^2 \ .$$

Linear regression

We are given a labelled data set $(\boldsymbol{x}^{(1)}, y^{(1)}), \ldots, (\boldsymbol{x}^{(n)}, y^{(n)}) \in \mathbb{R}^d \times \mathbb{R}$ so that

$$y^{(i)} = \ell^*(\boldsymbol{x}^{(i)}) + \epsilon^{(i)} \ ,$$

$\ell^*(\boldsymbol{x}) = \langle \theta^*, \boldsymbol{x} \rangle$ is an unknown linear function that we want to recover.

Assume that $\epsilon^{(i)}$ are independent noise variables.

Goal: Find a linear $\ell(\boldsymbol{x})$ minimizing

$$\mathrm{MSE}(\ell) = \frac{1}{n} \sum_{i=1}^{n} (\ell(\boldsymbol{x}^{(i)}) - \ell^*(\boldsymbol{x}^{(i)}))^2 \ .$$

We consider *fixed design* regression: we assume the $\boldsymbol{x}^{(i)}$ are fixed, and the only randomness is over the $\epsilon^{(i)}$.

The least squares estimator

Definition (Least squares estimator)

The least squares estimator, denoted $\widehat{\ell}^{\mathrm{LS}}$, is given by:

$$\widehat{\ell}^{\mathrm{LS}} \overset{def}{=} \underset{\substack{\ell \text{ linear}}}{\arg\min} \frac{1}{n} \sum_{i=1}^{n} (y^{(i)} - \ell(\boldsymbol{x}^{(i)}))^2 \ .$$

The least squares estimator

Definition (Least squares estimator)

The least squares estimator, denoted $\widehat{\ell}^{\mathrm{LS}}$, is given by:

$$\widehat{\ell}^{\mathrm{LS}} \stackrel{def}{=} \underset{\substack{\ell \text{ linear}}}{\arg\min} \frac{1}{n} \sum_{i=1}^{n} (y^{(i)} - \ell(\boldsymbol{x}^{(i)}))^2 \ .$$

The least squares fit simply the best fit linear function to the data.

The least squares estimator

Definition (Least squares estimator)

The least squares estimator, denoted $\widehat{\ell}^{\mathrm{LS}}$, is given by:

$$\widehat{\ell}^{\mathrm{LS}} \stackrel{def}{=} \underset{\substack{\ell \text{ linear}}}{\arg\min} \frac{1}{n} \sum_{i=1}^{n} (y^{(i)} - \ell(\boldsymbol{x}^{(i)}))^2 \ .$$

The least squares fit simply the best fit linear function to the data.

How well does it recover the ground truth ℓ^*?

The least squares estimator

Theorem

Let $\widehat{\ell}^{\mathrm{LS}}$ be as above. Suppose that $\epsilon^{(i)} \sim \mathcal{N}(0, \sigma^2)$. Then with high probability,

$$\mathrm{MSE}(\widehat{\ell}^{\mathrm{LS}}) = O\left(\sigma^2 \frac{d}{n}\right) .$$

Moreover, $\widehat{\ell}^{\mathrm{LS}}$ can be computed in time $O(nd^2)$.

The least squares estimator

Theorem

Let $\widehat{\ell}^{\mathrm{LS}}$ be as above. Suppose that $\epsilon^{(i)} \sim \mathcal{N}(0, \sigma^2)$. Then with high probability,

$$\mathrm{MSE}(\widehat{\ell}^{\mathrm{LS}}) = O\left(\sigma^2 \frac{d}{n}\right) .$$

Moreover, $\widehat{\ell}^{\mathrm{LS}}$ can be computed in time $O(nd^2)$.

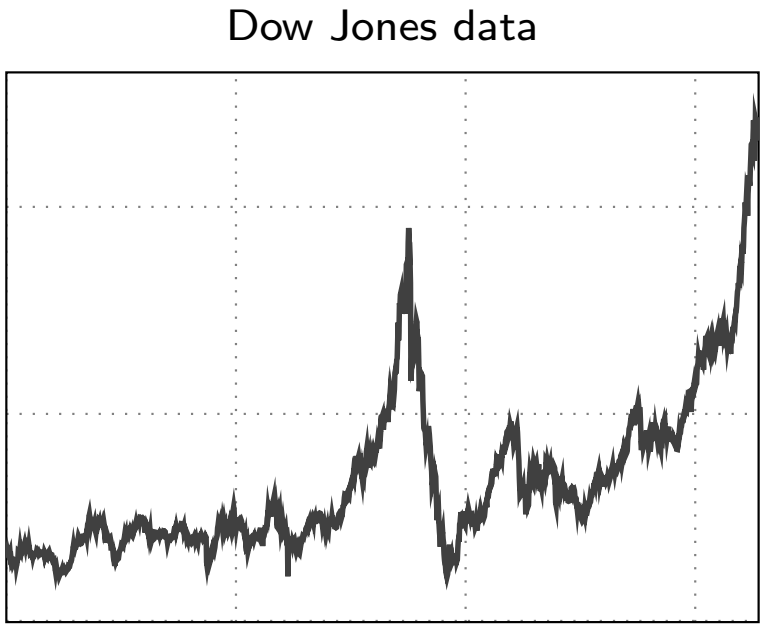
More recent work (see e.g. [CW13]) gets even faster theoretical runtimes.

Dealing with change

What if linear regression is insufficient?

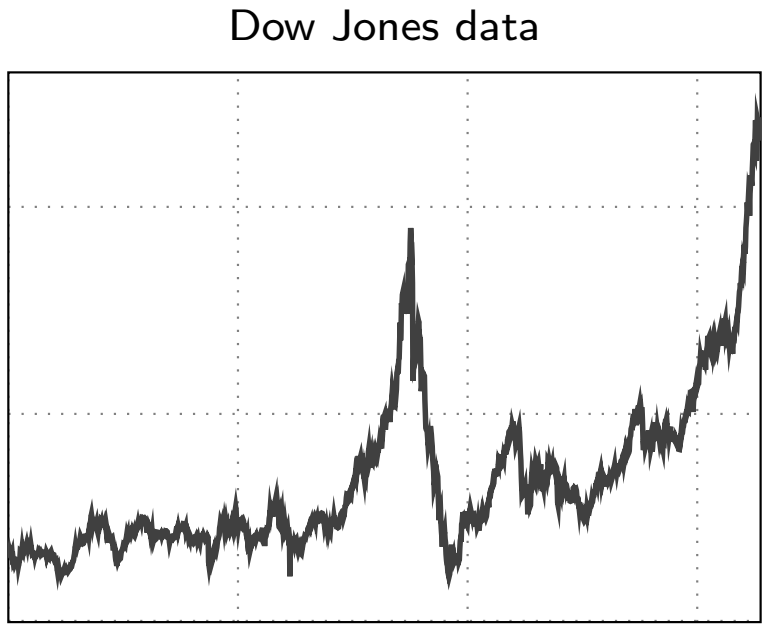
Dealing with change

What if linear regression is insufficient?



Dealing with change

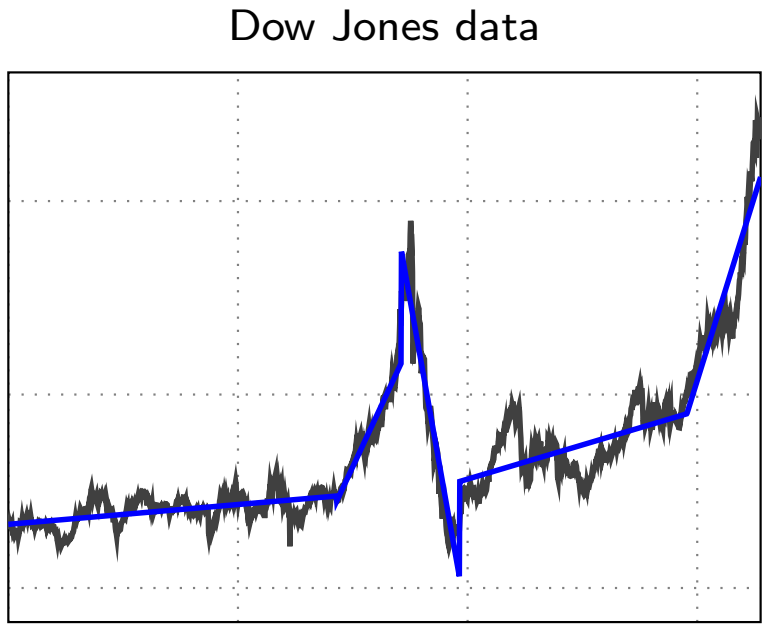
What if linear regression is insufficient?



Q: What if your model changes as a function of one of your variables?

Dealing with change

What if linear regression is insufficient?



Q: What if your model changes as a function of one of your variables?

A: Model it with a *piecewise linear* fit!

Segmented Regression

Segmented Regression

Definition (Piecewise linearity)

A function $f : \mathbb{R}^d \to \mathbb{R}$ is k-piecewise linear if there exists a partition of $\mathbb{R}$ into k intervals $I_1, \ldots, I_k$ so that for all j, the function f is linear restricted to the set of $\boldsymbol{x} \in \mathbb{R}^d$ so that $\boldsymbol{x}_1 \in I_j$.

Segmented Regression

Definition (Piecewise linearity)

A function $f : \mathbb{R}^d \to \mathbb{R}$ is k-piecewise linear if there exists a partition of $\mathbb{R}$ into k intervals $I_1, \ldots, I_k$ so that for all j, the function f is linear restricted to the set of $\boldsymbol{x} \in \mathbb{R}^d$ so that $\boldsymbol{x}_1 \in I_j$.

Segmented Regression [BP98, YP13]

Given a data set $(\boldsymbol{x}^{(1)}, y^{(1)}), \ldots, (\boldsymbol{x}^{(n)}, y^{(n)})$ so that

$$y^{(i)} = f(\boldsymbol{x}^{(i)}) + \epsilon^{(i)} \ ,$$

where $\epsilon^{(i)}$ are independent noise and f is k-piecewise linear, recover f in MSE.

Outline

- Introduction
- **The exact algorithm**
- Our algorithm
- Experiments

The k-piecewise linear LS estimator

Definition (Least squares estimator)

The k-piecewise linear least squares estimator, denoted $\widehat{f}_k^{\mathrm{LS}}$, is given by:

$$\widehat{f}_k^{\mathrm{LS}} \stackrel{\mathrm{def}}{=} \underset{f \ k\text{-piecewise linear}}{\arg\min} \frac{1}{n} \sum_{i=1}^{n} (y^{(i)} - f(\boldsymbol{x}^{(i)}))^2 \ .$$

The k-piecewise linear LS estimator

Definition (Least squares estimator)

The k-piecewise linear least squares estimator, denoted $\widehat{f}_k^{\mathrm{LS}}$, is given by:

$$\widehat{f}_k^{\mathrm{LS}} \stackrel{\mathrm{def}}{=} \underset{f \ k\text{-piecewise linear}}{\arg\min} \frac{1}{n}\sum_{i=1}^{n}(y^{(i)} - f(\boldsymbol{x}^{(i)}))^2 \ .$$

Theorem

Let $\widehat{f}_k^{\mathrm{LS}}$ be as above. Suppose that $\epsilon^{(i)} \sim \mathcal{N}(0, \sigma^2)$. Then with high probability,

$$\mathrm{MSE}(\widehat{f}_k^{\mathrm{LS}}) = O\left(\sigma^2 \frac{kd}{n}\right) \ .$$

Moreover, this rate is optimal.

The k-piecewise linear LS estimator, computationally

How fast can you compute this estimator?

The k-piecewise linear LS estimator, computationally

How fast can you compute this estimator?

Theorem (BP98)

There is a dynamic program for computing the k-piecewise linear LS estimator on n samples in d dimensions which runs in time $O(n^2(d^2 + k))$.

The k-piecewise linear LS estimator, computationally

How fast can you compute this estimator?

Theorem (BP98)

There is a dynamic program for computing the k-piecewise linear LS estimator on n samples in d dimensions which runs in time $O(n^2(d^2 + k))$.

So poly time...but quite slow as n gets large.

The k-piecewise linear LS estimator, computationally

How fast can you compute this estimator?

Theorem (BP98)

There is a dynamic program for computing the k-piecewise linear LS estimator on n samples in d dimensions which runs in time $O(n^2(d^2 + k))$.

So poly time...but quite slow as n gets large.

The algorithm took $\Theta(1)$ minutes to run for 10^4 samples, and $\Theta(1)$ hours to run for 10^5 samples.

Outline

- Introduction
- The exact algorithm
- **Our algorithm**
- Experiments

Our Results

Main Result (informally)

An algorithm for segmented regression which runs in time which is *linear* in n...

Our Results

Main Result (informally)

An algorithm for segmented regression which runs in time which is *linear* in n...but has a worse theoretical statistical guarantee.

Our Results

Main Result (informally)

An algorithm for segmented regression which runs in time which is *linear* in n...but has a worse theoretical statistical guarantee.

Formally...

Theorem

There is an $4k$-piecewise linear estimator $\widehat{f}$ which can be computed in time $O(n(d^2 + k))$ so that w.h.p.

$$\mathrm{MSE}(\widehat{f}) \leq \widetilde{O}\left(\sigma^2 \frac{kd}{n} + \sigma\sqrt{\frac{k}{n}}\right) .$$

Compare and contrast

Algorithm	Statistical Rate	Runtime

Compare and contrast

Algorithm	Statistical Rate	Runtime
DP		

Compare and contrast

Algorithm	Statistical Rate	Runtime
DP	$O\left(\sigma^2 \frac{kd}{n}\right)$	

Compare and contrast

Algorithm	Statistical Rate	Runtime
DP	$O\left(\sigma^2 \frac{kd}{n}\right)$	$O(n^2(d^2+k))$

Compare and contrast

Algorithm	Statistical Rate	Runtime
DP	$O\left(\sigma^2 \frac{kd}{n}\right)$	$O(n^2(d^2+k))$
Our Results		

Compare and contrast

Algorithm	Statistical Rate	Runtime
DP	$O\left(\sigma^2 \frac{kd}{n}\right)$	$O(n^2(d^2+k))$
Our Results		$O(n(d^2+k))$

Compare and contrast

Algorithm	Statistical Rate	Runtime
DP	$O\left(\sigma^2 \frac{kd}{n}\right)$	$O(n^2(d^2+k))$
Our Results	$\widetilde{O}\left(\sigma^2 \frac{kd}{n} + \sigma\sqrt{\frac{k}{n}}\right)$	$O(n(d^2+k))$

Compare and contrast

Algorithm	Statistical Rate	Runtime
DP	$O\left(\sigma^2 \frac{kd}{n}\right)$	$O(n^2(d^2+k))$
Our Results	$\widetilde{O}\left(\sigma^2 \frac{kd}{n} + \sigma\sqrt{\frac{k}{n}}\right)$	$O(n(d^2+k))$

Given enough data, how much time does it take to get some target accuracy ϵ?

Compare and contrast

Algorithm	Statistical Rate	Runtime
DP	$O\left(\sigma^2 \frac{kd}{n}\right)$	$O(n^2(d^2+k))$
Our Results	$\widetilde{O}\left(\sigma^2 \frac{kd}{n} + \sigma\sqrt{\frac{k}{n}}\right)$	$O(n(d^2+k))$

Given enough data, how much time does it take to get some target accuracy ϵ?

- **DP:** $O\left(\sigma^2 \frac{k^2 d^2}{\epsilon^2}(d^2+k)\right)$

Compare and contrast

Algorithm	Statistical Rate	Runtime
DP	$O\left(\sigma^2 \frac{kd}{n}\right)$	$O(n^2(d^2+k))$
Our Results	$\widetilde{O}\left(\sigma^2 \frac{kd}{n} + \sigma\sqrt{\frac{k}{n}}\right)$	$O(n(d^2+k))$

Given enough data, how much time does it take to get some target accuracy ϵ?

- **DP:** $O\left(\sigma^2 \frac{k^2 d^2}{\epsilon^2}(d^2+k)\right)$
- **Our Results**: $\widetilde{O}\left(\sigma^2 \frac{k}{\epsilon} \max\left(d, \frac{1}{\epsilon}\right)(d^2+k)\right)$

Compare and contrast

Algorithm	Statistical Rate	Runtime
DP	$O\left(\sigma^2 \frac{kd}{n}\right)$	$O(n^2(d^2+k))$
Our Results	$\widetilde{O}\left(\sigma^2 \frac{kd}{n} + \sigma\sqrt{\frac{k}{n}}\right)$	$O(n(d^2+k))$

Given enough data, how much time does it take to get some target accuracy ϵ?

- **DP:** $O\left(\sigma^2 \frac{k^2 d^2}{\epsilon^2}(d^2+k)\right)$
- **Our Results**: $\widetilde{O}\left(\sigma^2 \frac{k}{\epsilon} \max\left(d, \frac{1}{\epsilon}\right)(d^2+k)\right)$

Speedup: $\widetilde{O}(\min\left(\frac{kd}{\epsilon}, kd^2\right))$

The Greedy Merging Algorithm

Input: A labelled data set $(\boldsymbol{x}^{(1)}, y^{(1)}), \ldots (\boldsymbol{x}^{(n)}, y^{(n)})$.

The Greedy Merging Algorithm

Input: A labelled data set $(\boldsymbol{x}^{(1)}, y^{(1)}), \ldots (\boldsymbol{x}^{(n)}, y^{(n)})$.

- Sort them so that $\boldsymbol{x}_1^{(1)} \leq \boldsymbol{x}_1^{(2)} \leq \ldots \leq \boldsymbol{x}_1^{(n)}$.

The Greedy Merging Algorithm

Input: A labelled data set $(\boldsymbol{x}^{(1)}, y^{(1)}), \ldots (\boldsymbol{x}^{(n)}, y^{(n)})$.

- Sort them so that $\boldsymbol{x}_1^{(1)} \leq \boldsymbol{x}_1^{(2)} \leq \ldots \leq \boldsymbol{x}_1^{(n)}$.
- Let $\mathcal{I}_0 \leftarrow \{\{\boldsymbol{x}_1^{(1)}\}, \{\boldsymbol{x}_2^{(1)}\}, \ldots, \{\boldsymbol{x}_1^{(n)}\}\}$.

The Greedy Merging Algorithm

Input: A labelled data set $(\boldsymbol{x}^{(1)}, y^{(1)}), \ldots (\boldsymbol{x}^{(n)}, y^{(n)})$.

- Sort them so that $\boldsymbol{x}_1^{(1)} \leq \boldsymbol{x}_1^{(2)} \leq \ldots \leq \boldsymbol{x}_1^{(n)}$.
- Let $\mathcal{I}_0 \leftarrow \{\{\boldsymbol{x}_1^{(1)}\}, \{\boldsymbol{x}_2^{(1)}\}, \ldots, \{\boldsymbol{x}_1^{(n)}\}\}$.
- **While** $|\mathcal{I}_j| > 4k$:

The Greedy Merging Algorithm

Input: A labelled data set $(\boldsymbol{x}^{(1)}, y^{(1)}), \ldots (\boldsymbol{x}^{(n)}, y^{(n)})$.

- Sort them so that $\boldsymbol{x}_1^{(1)} \leq \boldsymbol{x}_1^{(2)} \leq \ldots \leq \boldsymbol{x}_1^{(n)}$.
- Let $\mathcal{I}_0 \leftarrow \{\{\boldsymbol{x}_1^{(1)}\}, \{\boldsymbol{x}_2^{(1)}\}, \ldots, \{\boldsymbol{x}_1^{(n)}\}\}$.
- **While** $|\mathcal{I}_j| > 4k$:
  - Let $\mathcal{I}_j = I_1, \ldots, I_s$.

The Greedy Merging Algorithm

Input: A labelled data set $(\boldsymbol{x}^{(1)}, y^{(1)}), \ldots (\boldsymbol{x}^{(n)}, y^{(n)})$.

- Sort them so that $\boldsymbol{x}_1^{(1)} \leq \boldsymbol{x}_1^{(2)} \leq \ldots \leq \boldsymbol{x}_1^{(n)}$.
- Let $\mathcal{I}_0 \leftarrow \{\{\boldsymbol{x}_1^{(1)}\}, \{\boldsymbol{x}_2^{(1)}\}, \ldots, \{\boldsymbol{x}_1^{(n)}\}\}$.
- **While** $|\mathcal{I}_j| > 4k$:
  - Let $\mathcal{I}_j = I_1, \ldots, I_s$.
  - Pair up consecutive intervals: $J_u = I_{2u-1} \cup I_{2u}, \ \forall u = 1, \ldots, s/2$.

The Greedy Merging Algorithm

Input: A labelled data set $(\boldsymbol{x}^{(1)}, y^{(1)}), \ldots (\boldsymbol{x}^{(n)}, y^{(n)})$.

- Sort them so that $\boldsymbol{x}_1^{(1)} \leq \boldsymbol{x}_1^{(2)} \leq \ldots \leq \boldsymbol{x}_1^{(n)}$.
- Let $\mathcal{I}_0 \leftarrow \{\{\boldsymbol{x}_1^{(1)}\}, \{\boldsymbol{x}_2^{(1)}\}, \ldots, \{\boldsymbol{x}_1^{(n)}\}\}$.
- **While** $|\mathcal{I}_j| > 4k$:
  - Let $\mathcal{I}_j = I_1, \ldots, I_s$.
  - Pair up consecutive intervals: $J_u = I_{2u-1} \cup I_{2u}, \ \forall u = 1, \ldots, s/2$.
  - For each J_u, compute the least squares fit for all data points in J_u, and an "error quantity" e_u.

The Greedy Merging Algorithm

Input: A labelled data set $(\boldsymbol{x}^{(1)}, y^{(1)}), \ldots (\boldsymbol{x}^{(n)}, y^{(n)})$.

- Sort them so that $\boldsymbol{x}_1^{(1)} \leq \boldsymbol{x}_1^{(2)} \leq \ldots \leq \boldsymbol{x}_1^{(n)}$.
- Let $\mathcal{I}_0 \leftarrow \{\{\boldsymbol{x}_1^{(1)}\}, \{\boldsymbol{x}_2^{(1)}\}, \ldots, \{\boldsymbol{x}_1^{(n)}\}\}$.
- **While** $|\mathcal{I}_j| > 4k$:
  - Let $\mathcal{I}_j = I_1, \ldots, I_s$.
  - Pair up consecutive intervals: $J_u = I_{2u-1} \cup I_{2u}, \ \forall u = 1, \ldots, s/2$.
  - For each J_u, compute the least squares fit for all data points in J_u, and an "error quantity" e_u.
  - Let $\mathcal{L}$ be the set of $2k$ u's with largest e_u

The Greedy Merging Algorithm

Input: A labelled data set $(\boldsymbol{x}^{(1)}, y^{(1)}), \ldots (\boldsymbol{x}^{(n)}, y^{(n)})$.

- Sort them so that $\boldsymbol{x}_1^{(1)} \leq \boldsymbol{x}_1^{(2)} \leq \ldots \leq \boldsymbol{x}_1^{(n)}$.
- Let $\mathcal{I}_0 \leftarrow \{\{\boldsymbol{x}_1^{(1)}\}, \{\boldsymbol{x}_2^{(1)}\}, \ldots, \{\boldsymbol{x}_1^{(n)}\}\}$.
- **While** $|\mathcal{I}_j| > 4k$:
  - Let $\mathcal{I}_j = I_1, \ldots, I_s$.
  - Pair up consecutive intervals: $J_u = I_{2u-1} \cup I_{2u}, \ \forall u = 1, \ldots, s/2$.
  - For each J_u, compute the least squares fit for all data points in J_u, and an "error quantity" e_u.
  - Let $\mathcal{L}$ be the set of $2k$ u's with largest e_u
  - **For** $u = 1, \ldots, s/2$:

The Greedy Merging Algorithm

Input: A labelled data set $(\boldsymbol{x}^{(1)}, y^{(1)}), \ldots (\boldsymbol{x}^{(n)}, y^{(n)})$.

- Sort them so that $\boldsymbol{x}_1^{(1)} \leq \boldsymbol{x}_1^{(2)} \leq \ldots \leq \boldsymbol{x}_1^{(n)}$.
- Let $\mathcal{I}_0 \leftarrow \{\{\boldsymbol{x}_1^{(1)}\}, \{\boldsymbol{x}_2^{(1)}\}, \ldots, \{\boldsymbol{x}_1^{(n)}\}\}$.
- **While** $|\mathcal{I}_j| > 4k$:
  - Let $\mathcal{I}_j = I_1, \ldots, I_s$.
  - Pair up consecutive intervals: $J_u = I_{2u-1} \cup I_{2u}, \ \forall u = 1, \ldots, s/2$.
  - For each J_u, compute the least squares fit for all data points in J_u, and an "error quantity" e_u.
  - Let $\mathcal{L}$ be the set of $2k$ u's with largest e_u
  - **For** $u = 1, \ldots, s/2$:
    - **If** $u \notin \mathcal{L}$, include $I_{2u-1} \cup I_{2u}$ in $\mathcal{I}_{j+1}$

The Greedy Merging Algorithm

Input: A labelled data set $(\boldsymbol{x}^{(1)}, y^{(1)}), \ldots (\boldsymbol{x}^{(n)}, y^{(n)})$.

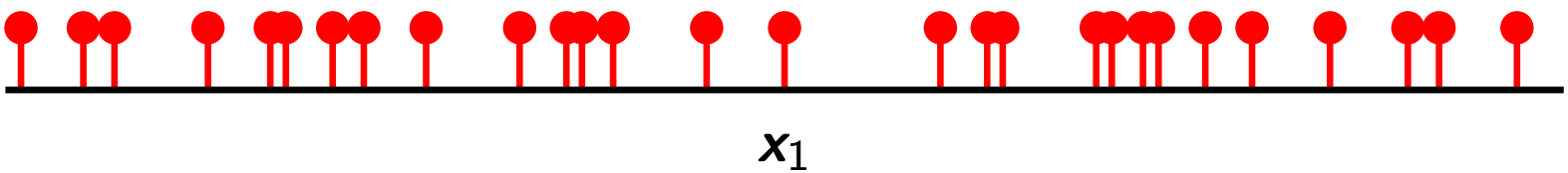
- Sort them so that $\boldsymbol{x}_1^{(1)} \leq \boldsymbol{x}_1^{(2)} \leq \ldots \leq \boldsymbol{x}_1^{(n)}$.
- Let $\mathcal{I}_0 \leftarrow \{\{\boldsymbol{x}_1^{(1)}\}, \{\boldsymbol{x}_2^{(1)}\}, \ldots, \{\boldsymbol{x}_1^{(n)}\}\}$.
- **While** $|\mathcal{I}_j| > 4k$:
  - Let $\mathcal{I}_j = I_1, \ldots, I_s$.
  - Pair up consecutive intervals: $J_u = I_{2u-1} \cup I_{2u}, \ \forall u = 1, \ldots, s/2$.
  - For each J_u, compute the least squares fit for all data points in J_u, and an "error quantity" e_u.
  - Let $\mathcal{L}$ be the set of $2k$ u's with largest e_u
  - **For** $u = 1, \ldots, s/2$:
    - **If** $u \notin \mathcal{L}$, include $I_{2u-1} \cup I_{2u}$ in $\mathcal{I}_{j+1}$
    - **Else** if $u \in \mathcal{L}$ include I_{2u-1} and I_{2u} in $\mathcal{I}_{j+1}$.

The Greedy Merging Algorithm

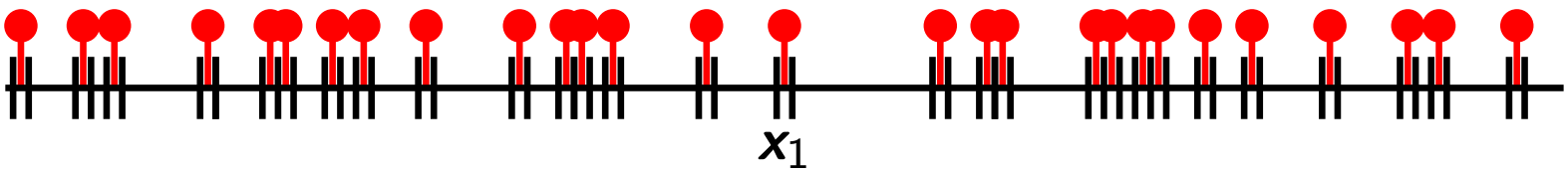
Input: A labelled data set $(\boldsymbol{x}^{(1)}, y^{(1)}), \ldots (\boldsymbol{x}^{(n)}, y^{(n)})$.

- Sort them so that $\boldsymbol{x}_1^{(1)} \leq \boldsymbol{x}_1^{(2)} \leq \ldots \leq \boldsymbol{x}_1^{(n)}$.
- Let $\mathcal{I}_0 \leftarrow \{\{\boldsymbol{x}_1^{(1)}\}, \{\boldsymbol{x}_2^{(1)}\}, \ldots, \{\boldsymbol{x}_1^{(n)}\}\}$.
- **While** $|\mathcal{I}_j| > 4k$:
  - Let $\mathcal{I}_j = I_1, \ldots, I_s$.
  - Pair up consecutive intervals: $J_u = I_{2u-1} \cup I_{2u}, \ \forall u = 1, \ldots, s/2$.
  - For each J_u, compute the least squares fit for all data points in J_u, and an "error quantity" e_u.
  - Let $\mathcal{L}$ be the set of $2k$ u's with largest e_u
  - **For** $u = 1, \ldots, s/2$:
    - **If** $u \notin \mathcal{L}$, include $I_{2u-1} \cup I_{2u}$ in $\mathcal{I}_{j+1}$
    - **Else** if $u \in \mathcal{L}$ include I_{2u-1} and I_{2u} in $\mathcal{I}_{j+1}$.
- **Output** The linear least squares fit over each interval in $\mathcal{I}_j$

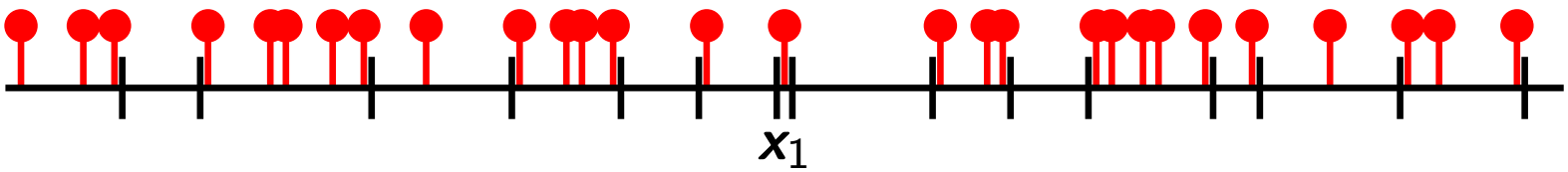
Example: $k = 2$



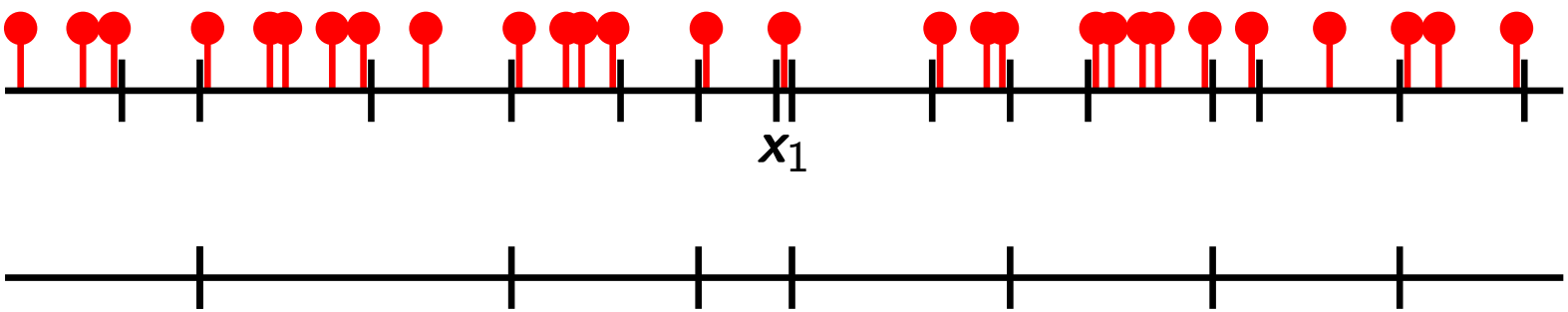
Example: $k = 2$



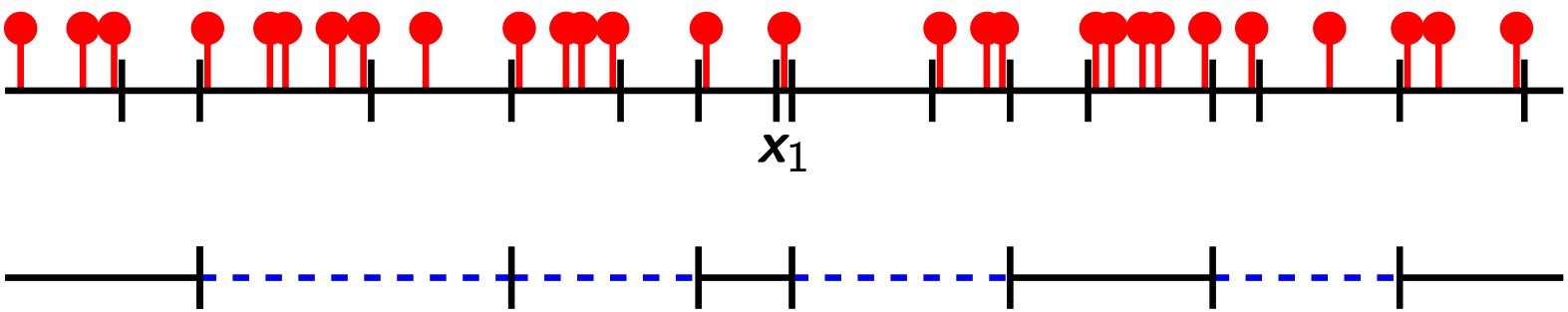
Example: $k = 2$



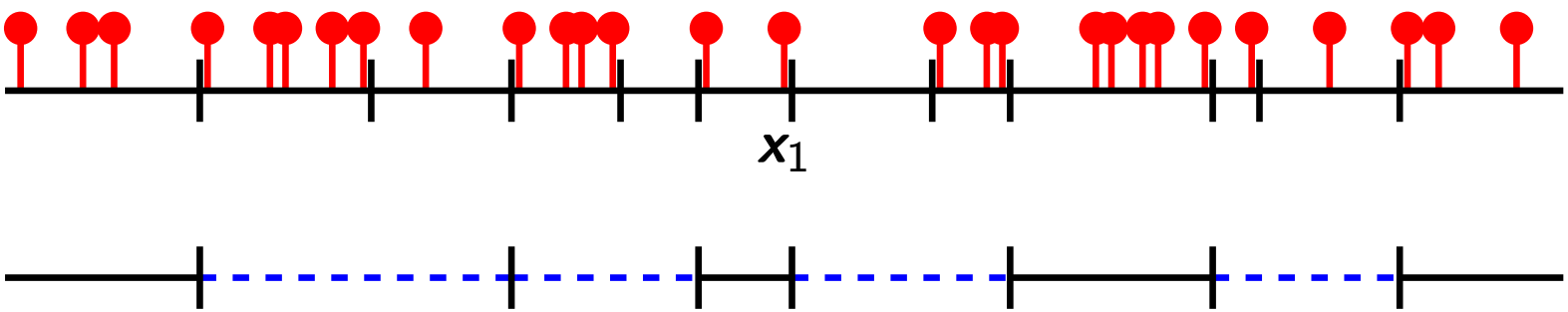
Example: $k = 2$



Example: $k = 2$



Example: $k = 2$



Remarks

Remarks

- Algorithmically similar to an algorithm due to [ADHLS15] for histogram approximation—however analysis is quite different and more involved here.

Remarks

- Algorithmically similar to an algorithm due to [ADHLS15] for histogram approximation—however analysis is quite different and more involved here.

- Can get a smooth tradeoff between runtime and number of pieces—see paper for details.

Remarks

- Algorithmically similar to an algorithm due to [ADHLS15] for histogram approximation—however analysis is quite different and more involved here.
- Can get a smooth tradeoff between runtime and number of pieces—see paper for details.
- The error rule we use requires knowledge of σ^2—we also give a similar algorithm which (up to log factors) matches the same guarantees as before which requires no such knowledge.

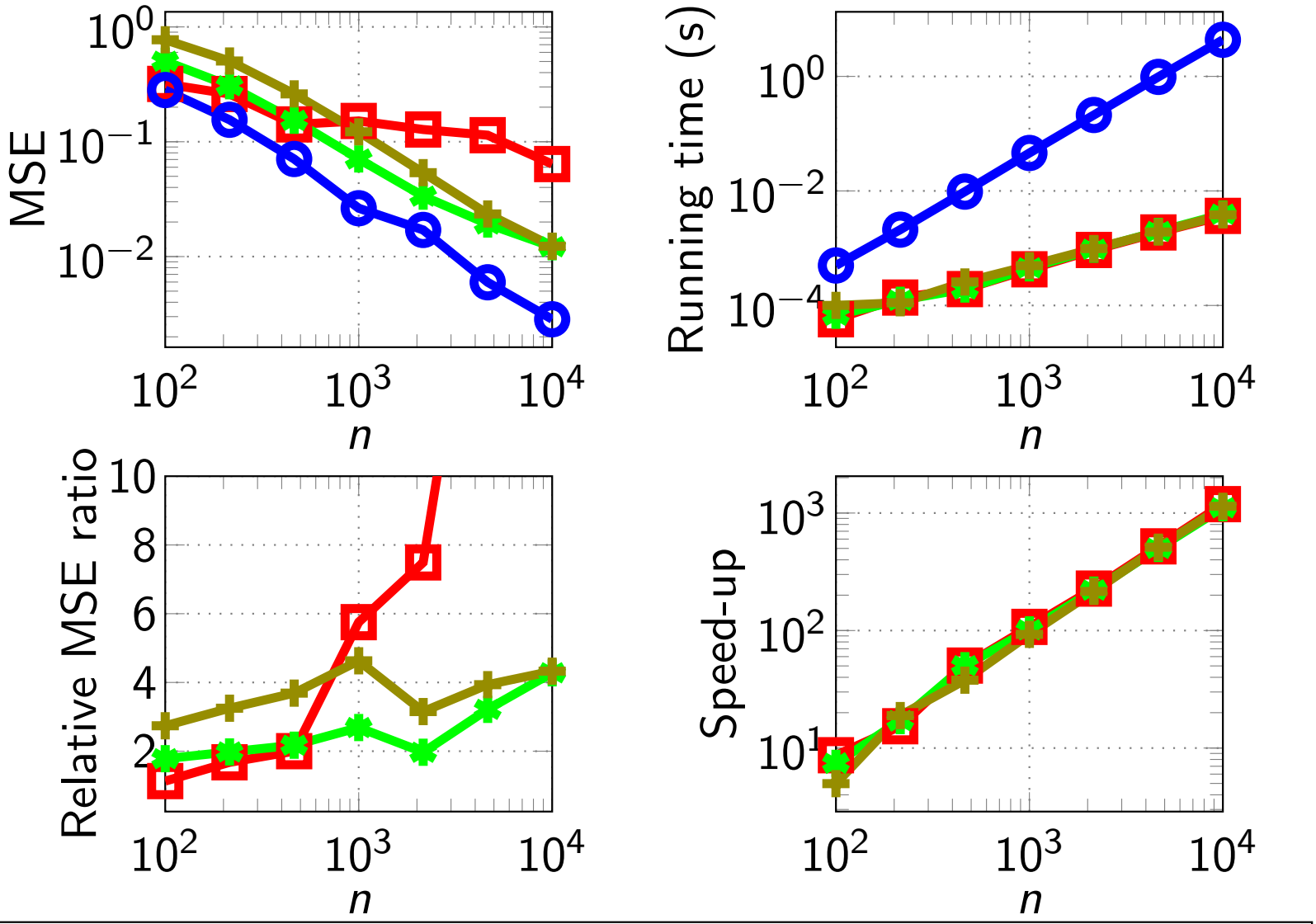
Remarks

- Algorithmically similar to an algorithm due to [ADHLS15] for histogram approximation—however analysis is quite different and more involved here.
- Can get a smooth tradeoff between runtime and number of pieces—see paper for details.
- The error rule we use requires knowledge of σ^2—we also give a similar algorithm which (up to log factors) matches the same guarantees as before which requires no such knowledge.
- Can show that our algorithms are robust to model misspecification.

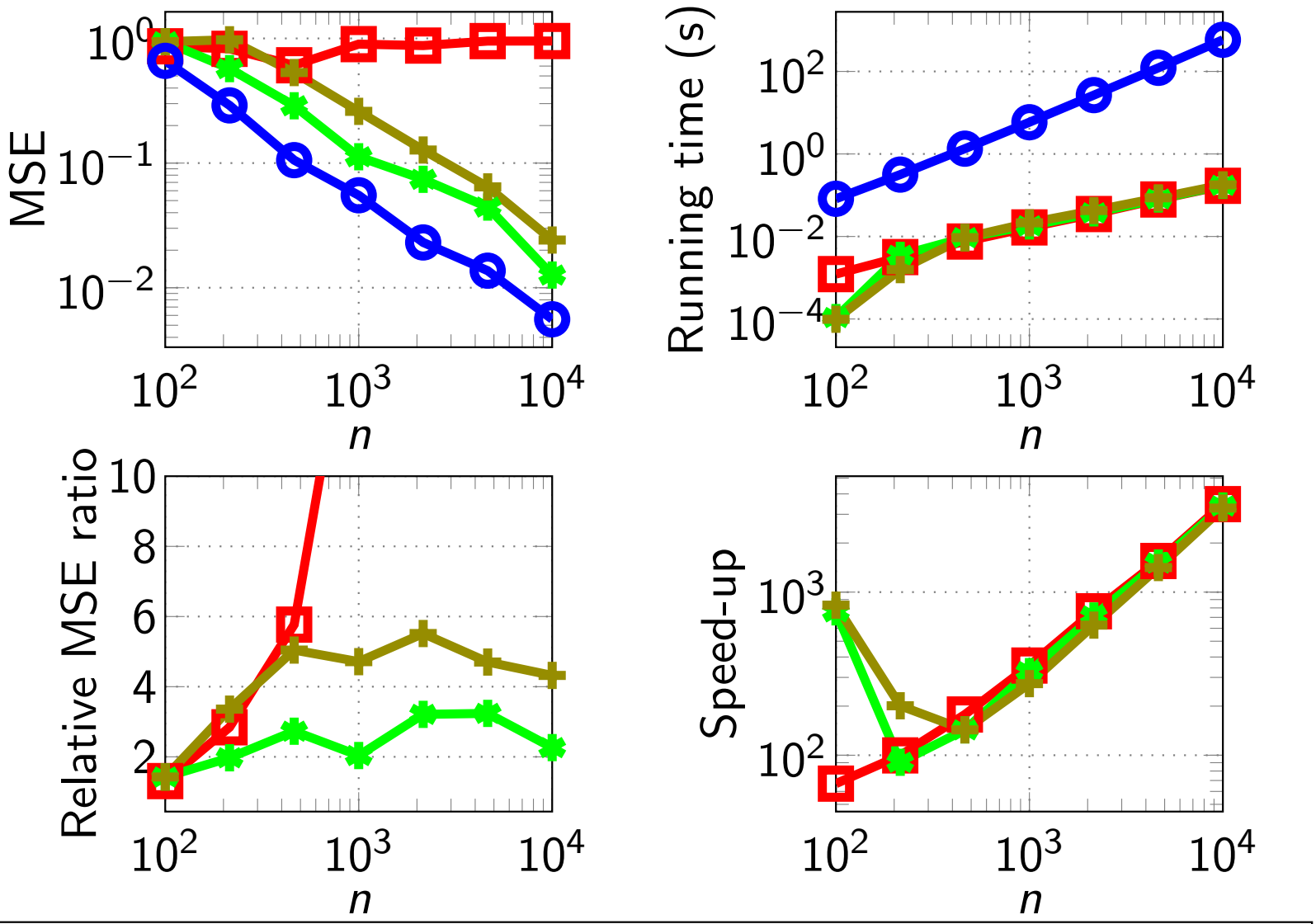
Outline

- Introduction
- The exact algorithm
- Our algorithm
- **Experiments**

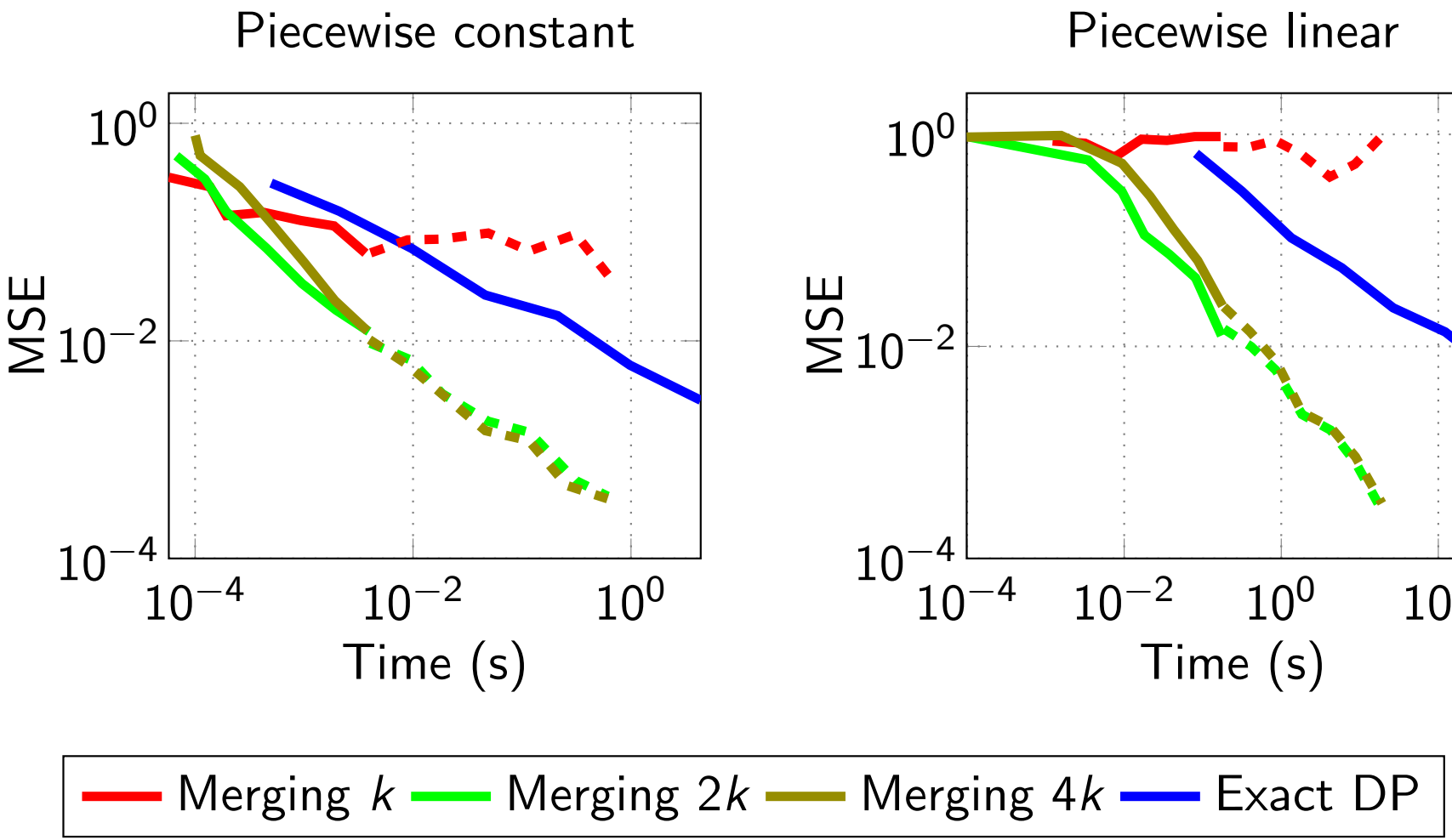
Experiments: piecewise constant



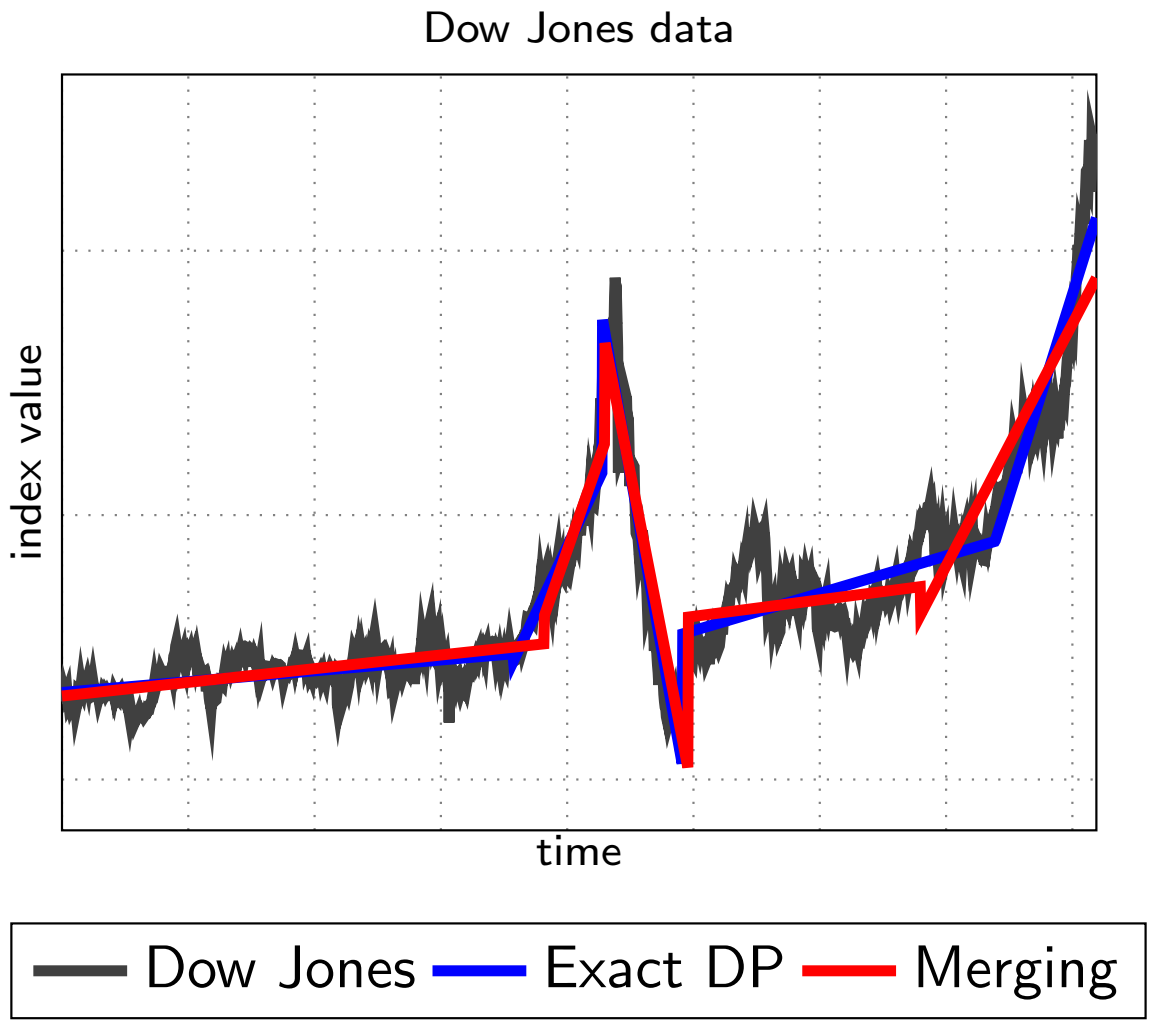
Experiments: piecewise linear



Experiments: time vs error trade-off



Experiments: real data



Experiments

Experiments

- Our algorithm performs $1000\times$ faster with $n = 10^4$

Experiments

- Our algorithm performs $1000\times$ faster with $n = 10^4$
- Our algorithm's MSE on synthetic data was $2 - 4$ times worse than the DP's

Experiments

- Our algorithm performs $1000\times$ faster with $n = 10^4$
- Our algorithm's MSE on synthetic data was $2 - 4$ times worse than the DP's
- Given enough data, we get the same MSE $100\times$ faster

Conclusions

Conclusions

- When is it worth it to trade statistical effectiveness for algorithmic efficiency?

Conclusions

- When is it worth it to trade statistical effectiveness for algorithmic efficiency?
- We give an algorithm for segmented regression that gets a worse theoretical MSE, but a much faster runtime

Conclusions

- When is it worth it to trade statistical effectiveness for algorithmic efficiency?
- We give an algorithm for segmented regression that gets a worse theoretical MSE, but a much faster runtime
- Experimentally our algorithm has slightly worse MSE, but runs 1000$\times$ faster.

Conclusions

- When is it worth it to trade statistical effectiveness for algorithmic efficiency?
- We give an algorithm for segmented regression that gets a worse theoretical MSE, but a much faster runtime
- Experimentally our algorithm has slightly worse MSE, but runs 1000$\times$ faster.
- **Open Question:** Is this tradeoff necessary?

Conclusions

- When is it worth it to trade statistical effectiveness for algorithmic efficiency?
- We give an algorithm for segmented regression that gets a worse theoretical MSE, but a much faster runtime
- Experimentally our algorithm has slightly worse MSE, but runs 1000$\times$ faster.
- **Open Question:** Is this tradeoff necessary?

Thank you!