The Complexity of Dynamic Least-Squares Regression

Shunhua Jiang∗  Binghui Peng∗  Omri Weinstein†
FOCS 2023

∗Columbia University  †Columbia University & Hebrew University
Least squares regression

- **Problem:** \( \min_{x \in \mathbb{R}^d} \| Ax - b \|_2 \)
- **Applications** in high-dimensional statistical inference, signal processing, machine learning, etc.
- **Exact solution** (Normal equation): \( x^* = (A^\top A)^{-1} A^\top b \)
  - **Time complexity:** \( O(nd^{\omega-1}) \)
  - Still too slow for many modern data-analysis applications.
- \( \epsilon \)-**approximate solution:** \( \| Ax - b \|_2 \leq (1 + \epsilon) \min_{x' \in \mathbb{R}^d} \| Ax' - b \|_2 \)
  - “Sketch and solve” paradigm [Woo14]
  - **Time complexity:** \( \tilde{O}((\text{nnz}(A) + d^\omega) \log(1/\epsilon)) \) [CW17]
**Dynamic least squares regression**

- **Problem:** Dynamically maintain an $\epsilon$-approximate LSR solution
  \[
  \min_{x \in \mathbb{R}^d} \| A^{(i)} x - b^{(i)} \|_2,
  \]
  under insertion or deletion of rows $a^{(i)} \in \mathbb{R}^d$ and labels $\beta^{(i)} \in \mathbb{R}$.
  - Goal: minimize **amortized update time**.
  - In total $n$ iterations, think of $n = \text{poly}(d)$.

- Models dynamic data applications, e.g., continual ML.

- **Incremental vs Fully dynamic**
  - Incremental: Only insertions of rows.
  - Fully dynamic: Both insertions and deletions of rows.

- **Oblivious updates vs Adaptive updates**
  - Oblivious updates: The sequence of updates are fixed in the beginning.
  - Adaptive updates: The next update is generated based on the previous outputs.
Algorithms for dynamic least squares regression

- **Exact solution**: Update the normal equation \( x^{*,(i)} = (A^{(i)^\top}A^{(i)})^{-1}A^{(i)^\top}b^{(i)} \) using Woodbury identity. (Kalman filters [Kal60])
  - Works for fully dynamic and adaptive updates.
  - Time per update: \( O(d^2) \).

- **Online row sampling** [CMP20]: Maintain an \( \epsilon \)-approximate solution by sampling \( O(d \log \kappa/\epsilon^2) \) number of rows, where \( \kappa := \frac{\sigma_{\text{max}}(A^{(n)})}{\sigma_{\text{min}}(A^{(0)})} \).
  - Works for incremental and oblivious updates.
  - Time per update: \( O(d^2) \) (to compute sampling probability).

- **Adaptive online row sampling** [BHM+21]: Sample \( O(d^2\kappa \log \kappa/\epsilon^2) \) number of rows, where \( \kappa := \frac{\sigma_{\text{max}}(A^{(n)})}{\sigma_{\text{min}}(A^{(0)})} \).
  - Works for incremental and adaptive updates.
  - Time per update: \( O(d^2) \).

- **Question**: Can we achieve \( O(d) \) time per update / \( O(nd) \) total time?
Our results: Upper bound

**Theorem (Upper bound).** There is a dynamic data structure that maintains an $\epsilon$-approximate LSR solution under *oblivious incremental* updates, with total time $\tilde{O}(nd + d^3 \text{poly}(\epsilon^{-1}))$. The data structure can be made to work against *adaptive incremental* updates with total time $\tilde{O}(nd + d^5 \text{poly}(\epsilon^{-1}))$.

- When $n \gg d$ and $\epsilon$ is a small constant, the amortized cost per iteration is $\tilde{O}(d)$.
- The $nd$ term is in fact $\text{nnz}(A^{(n)})$.
- For *adaptive* incremental updates, we improve the number of sampled rows from $O(d^2 \kappa \log \kappa / \epsilon^2)$ [BHM+21] to $O(d^2 \log^2 \kappa / \epsilon^2)$.
- **Question:** Can we improve $\text{poly}(\epsilon^{-1})$ dependence to $\log(\epsilon^{-1})$ as the static case?
- **Question:** Algorithms for fully dynamic updates?
Theorem (Lower bound). Under the OMv conjecture: [HKNS15]

- **High vs low accuracy.** Any dynamic data structure that maintains an \( \epsilon = 1/\text{poly}(n) \)-approximate LSR solution under *oblivious incremental* updates requires \( \Omega(d^{2-o(1)}) \) amortized cost per iteration.

- **Fully vs partially dynamic.** If the data structure supports *adaptive fully dynamic* updates, then maintaining \( \epsilon = 0.01 \)-approximate LSR solution requires \( \Omega(d^{2-o(1)}) \) amortized cost per iteration.
  - Impossible to improve \( \text{poly}(\epsilon^{-1}) \) dependence to \( \log(\epsilon^{-1}) \).
  - Impossible to make the algorithm work for fully dynamic updates.
I. Upper Bound: Incremental Oblivious Setting
**Exact solution for dynamic LSR**

- **Notations**: In the $i$-th iteration, given a new row $a^{(i)} \in \mathbb{R}^d$ and a new label $\beta^{(i)} \in \mathbb{R}$, solve for
\[
\min_{x \in \mathbb{R}^d} \| A^{(i)} x - b^{(i)} \|_2.
\]

- **Exact solution** (Kalman filters [Kal60]): Compute $x^{*,(i)} = (A^{(i)\top} A^{(i)})^{-1} A^{(i)\top} b^{(i)}$.
  - Inverse $\left( A^{(i)\top} A^{(i)} \right)^{-1} = \left( A^{(i-1)\top} A^{(i-1)} + a^{(i)} a^{(i)\top} \right)^{-1}$.
  - Woodbury identity: $\left( M + a^{(i)} a^{(i)\top} \right)^{-1} = M^{-1} - \frac{M^{-1} a^{(i)} a^{(i)\top} M^{-1}}{1 + a^{(i)\top} a^{(i)}}$.

- Time per update: $O(d^2)$. 

\[
\begin{bmatrix}
M \\
\end{bmatrix}
+ \begin{bmatrix} a \\
\end{bmatrix} \begin{bmatrix} a^\top \\
\end{bmatrix}^{-1}
= \begin{bmatrix} M^{-1} \\
\end{bmatrix}
- \frac{1}{1 + a^\top a} \begin{bmatrix} M^{-1} \\
\end{bmatrix} \begin{bmatrix} a \\
\end{bmatrix} \begin{bmatrix} a^\top \\
\end{bmatrix} \begin{bmatrix} M^{-1} \\
\end{bmatrix}
\]
Subspace embedding and approximate LSR

- **Subspace embedding** (See survey [Woo14]):
  Given a matrix $A \in \mathbb{R}^{n \times d}$, matrix $S \in \mathbb{R}^{s \times n}$ is a $(1 \pm \epsilon)$ subspace embedding for $A$ if
  $$\|SAx\|_2 = (1 \pm \epsilon)\|Ax\|_2$$ for all $x$.

- **Approx LSR**: Let $S$ be a $(1 \pm \epsilon)$ subspace embedding of matrix $[A, b]$.
  $$x' := \arg \min_{x \in \mathbb{R}^d} \|SAx - Sb\|_2$$
  is an $O(\epsilon)$-approximate solution for the original problem:
  $$\|Ax' - b\|_2 \leq (1 + \epsilon) \min_{x \in \mathbb{R}^d} \|Ax - b\|_2.$$

- Subspace embedding technique that is easy to dynamize: leverage score sampling
Leverage score sampling

- **Leverage scores**: For a fixed matrix $A$, the leverage score of its $i$-th row $a_i$ is
  \[
  \tau_i(A) := a_i^\top (A^\top A)^{-1} a_i.
  \]

  Diagonal entries of the projection matrix $A(A^\top A)^{-1} A^\top$.
- Measures how important the row $a_i$ is for the row space of $A$.
  - If $\tau_i(A) = 1$: removing row $i$ will decrease the rank of $A$ by 1.
  - If all rows are the same, they all have $\tau_i(A) = d/n$.
- **Main properties**: (i) $0 \leq \tau_i(A) \leq 1$. (ii) $\sum_{i=1}^n \tau_i(A) = d$.
- **Leverage score sampling**: Sample the $i$-th row with probability $p_i = \tau_i(A)/\epsilon^2$.
  Let $D_{ii} = 1/\sqrt{p_i}$ if the $i$-th row is sampled, and 0 otherwise. Then with high probability $D$ is a $(1 \pm \epsilon)$ subspace embedding for $A$.
- In expectation sample $\sum_{i=1}^n p_i = O(d/\epsilon^2)$ rows.
Online leverage score sampling [CMP20]

- Online leverage scores:
  \[ \tau_i := (a(i))^\top ((A(i-1))^\top A(i-1))^{-1} a(i) \]

- Overestimates: \( \bar{\tau}_i \geq \tau_i \) since \( (A(i-1))^\top A(i-1) \preceq (A(n))^\top A(n) \).

- Online leverage score sampling: When the \( i \)-th row arrives, sample it with probability \( p_i = \bar{\tau}_i / \epsilon^2 \). Let \( D_{ii} = 1/\sqrt{p_i} \) if the \( i \)-th row is sampled, and 0 otherwise. Then whp \( D \) is a \((1 \pm \epsilon)\) subspace embedding for \( A(i) \).

- Sum of online leverage scores: \( \sum_{i=1}^{n} \tau_i \leq d \log(d \kappa) \), where \( \kappa := \frac{\sigma_{\max}(A(n))}{\sigma_{\min}(A(0))} \).
  - Fact: \( \log \det(M + aa^\top) \geq \log \det(M) + a^\top M^{-1} a \).
  - Apply this fact to the rows:
    \[ \log \det((A(n))^\top A(n)) \geq \log \det((A(n-1))^\top A(n-1)) + \tau_n \geq \cdots \geq \log \det((A(0))^\top A(0)) + \sum_{i=1}^{n} \tau_i \]

- In expectation sample \( \sum_{i=1}^{n} p_i = \tilde{O}(d \log(\kappa) / \epsilon^2) \) rows.
Algorithm for oblivious updates

**Algorithm**: We maintain a subsampled matrix $\tilde{A} = DA^{(i)}$. In each iteration:

- When $a^{(i)}$ arrives, compute $\tau_i = a^{(i)\top} \cdot (\tilde{A}^\top \tilde{A})^{-1} \cdot a^{(i)}$. \(1\)
- Flip a coin with probability $p_i = \tau_i / \epsilon^2$:
  - If 1: Add $a^{(i)}/\sqrt{p_i}$ as a new row to $\tilde{A}$. Update $(\tilde{A}^\top \tilde{A})^{-1}$ and solution. \(2\)
  - If 0: Ignore $a^{(i)}$. Output the same solution.

**Update time** \(2\):

- One update takes $O(d^2)$ time by using Woodbury identity.
- The total number of updates is $\sum_{i=1}^n \tau_i / \epsilon^2 = \tilde{O}(d \log(\kappa) / \epsilon^2)$.
- Total time is $\tilde{O}(d^3 \log(\kappa) / \epsilon^2)$.
- Amortized cost is $d^{o(1)}$ when $n \gg d$. 

10
Computing leverage scores more efficiently

• Recall: We want to compute $\bar{\tau}_i = a^{(i)\top}(\tilde{A}^\top\tilde{A})^{-1}a^{(i)}$ in each iteration. Direct computation takes $O(d^2)$ time in [CMP20].

• **Johnson-Lindenstrauss lemma**: There exists JL matrix $J$ that compresses dimension from $d$ to $O(\log n)$ and guarantees $\|Jx\|_2^2 \approx 0.01 \|x\|_2^2$ for fixed $n$ vectors.

• $a^\top \cdot (A^\top A)^{-1} \cdot a = \|A(A^\top A)^{-1} \cdot a\|_2^2$. [SS08].

• The algorithm also maintains $J \cdot \tilde{A}(\tilde{A}^\top \tilde{A})^{-1}$.

• We have $\bar{\tau}_i = \|\tilde{A}(\tilde{A}^\top \tilde{A})^{-1} \cdot a^{(i)}\|_2^2 \approx 0.01 \|J\tilde{A}(\tilde{A}^\top \tilde{A})^{-1} \cdot a^{(i)}\|_2^2$

\[ \begin{bmatrix} J \\ \tilde{A}(\tilde{A}^\top \tilde{A})^{-1} \end{bmatrix} a^{(i)} \approx \begin{bmatrix} \tilde{A}(\tilde{A}^\top \tilde{A})^{-1} \\ a^{(i)} \end{bmatrix} \|
\]

• This estimate can be computed in $O(d \log n)$ time.

$\implies$ Total time is $O(nd \log n)$. 

Theorem (Upper bound in oblivious setting). There is a dynamic data structure that maintains an $\epsilon$-approximate LSR solution under oblivious incremental updates, with total time $O(nd \log n + d^3 \text{poly}(\epsilon^{-1}))$. 
II. Upper Bound: Incremental Adaptive Setting
Adaptive updates

- Adaptive updates are inherent in many iterative algorithms.
- To make our algorithm work against adaptive updates:
  - Make JL trick work against adaptive updates.
    - Make the JL estimate an over-estimate.
    - Renew the JL sketch whenever a row is sampled.
  - Make online leverage score sampling work against adaptive updates.
Proof of oblivious leverage score sampling

- **Leverage score sampling**: Sample the $i$-th row with probability $p_i = \tau_i(A)/\epsilon^2$. Let $D_{ii} = 1/\sqrt{p_i}$ if the $i$-th row is sampled, and 0 otherwise. Then whp $D$ is a $(1 \pm \epsilon)$ subspace embedding for $A$.

- **Matrix Chernoff bound**: Given independently random PSD matrices $X_1, \cdots, X_n \in \mathbb{R}^{d \times d}$ s.t. $X_i \preceq R \cdot I$. Let $W = \mathbb{E} \left[ \sum_{i=1}^{n} X_i \right]$. Then

  \[
  \Pr[\lambda_{\min}(\sum_{i=1}^{n} X_i) \leq (1 - \epsilon)\lambda_{\min}(W)] \leq d \cdot 2^{-e^2\lambda_{\min}(W)/R},
  \]

  \[
  \Pr[\lambda_{\max}(\sum_{i=1}^{n} X_i) \geq (1 + \epsilon)\lambda_{\max}(W)] \leq d \cdot 2^{-e^2\lambda_{\max}(W)/R}.
  \]

- **Proof of leverage score sampling**: Define $X_i := \begin{cases} \frac{1}{p_i} \cdot a^{(i)}(a^{(i)})^\top & \text{w.p. } p_i \\ 0 & \text{otherwise} \end{cases}$. Apply Matrix Chernoff bound to scaled version: $\overline{X}_i = W^{-1/2}X_i W^{-1/2}$.
Adaptive online leverage score sampling

- **Adaptive Matrix Chernoff bound.** Given adaptive random PSD matrices $X_1, \ldots, X_n \in \mathbb{R}^{d \times d}$ s.t. $X_i \preceq R \cdot I$. Let $W = \sum_{i=1}^{n} \mathbb{E}[X_i | X_1, \ldots, X_{i-1}]$. Then we have that for any $\mu$:

$$\Pr[\lambda_{\text{min}}(\sum_{i=1}^{n} X_i) \leq (1 - \epsilon)\mu \text{ and } \lambda_{\text{min}}(W) \geq \mu] \leq d \cdot 2^{-2\epsilon^2 \mu / R},$$

$$\Pr[\lambda_{\text{max}}(\sum_{i=1}^{n} X_i) \geq (1 + \epsilon)\mu \text{ and } \lambda_{\text{max}}(W) \leq \mu] \leq d \cdot 2^{-2\epsilon^2 \mu / R}.$$  

- $W$ is a random variable.
- Cannot use scaled version $\overline{X}_i = W^{-1/2}X_iW^{-1/2}$ anymore!
- By “guessing” the matrix $W$, and use a union bound over all “guesses”, we can prove $\epsilon$-approximation when $p_i = C \cdot \overline{\tau}_i / \epsilon^2$, where $C = \tilde{O}(d^2 \log(\kappa))$.
- Using scalar concentration bounds, only lose a factor of $C = \tilde{O}(d \log(\kappa))$. 

Lemma (Adaptive online leverage score sampling)

Let $a^{(1)}, \ldots, a^{(n)}$ be a sequence of adaptive updates. Sample the $i$-th row with probability $p_i = C \cdot \tau_i / \epsilon^2$, where $C = \tilde{O}(d \log(\kappa))$. Let $D_{ii} = 1 / \sqrt{p_i}$ if the $i$-th row is sampled, and 0 otherwise. Then whp $D$ is a $(1 \pm \epsilon)$ subspace embedding for $A$.

Proof ideas of [BHM+21]

- Instead of proving $DA \approx_\epsilon A$, prove the scalar case that $\|DAv\|_2 \approx_\epsilon \|Av\|_2$
- Need to prove this for all vector $v$'s in an $\epsilon$-net of size $(\kappa / \epsilon) \tilde{O}(d)$.
- Need $\delta < (\epsilon / \kappa) \tilde{O}(d)$ to use union bound.

$$\implies$$ Lose a factor of $d \log(\kappa)$ in $\log \frac{1}{\delta}$. 
Proof ideas of $[\text{BHM}^+21]$ (continued)

- Define $x_i := (D_{ii}^2 - 1) \cdot \mathbf{v}^\top a(i)(a(i))^\top \mathbf{v}$.
- Goal is to prove $|\sum_{i=1}^n x_i| \leq \epsilon \cdot \|A(n)\mathbf{v}\|_2^2$.
- Use concentration bound for scalar adaptive sequences: **Freedman's inequality** (simplified for talk). Let $x_1, \ldots, x_n \in \mathbb{R}$ be an adaptive sequence such that $\mathbb{E}[x_i \mid x_1, \ldots, x_{i-1}] = 0$, and $|x_i| \leq R$. Then for any $\mu$,

\[
\Pr[|\sum_{i=1}^n x_i| \geq \mu] \leq e^{-\mu/R}.
\]

- Would like to set $\mu = \epsilon \cdot \|A(n)\mathbf{v}\|_2^2$. However, $\|A(n)\mathbf{v}\|_2^2$ is a random variable!
- $[\text{BHM}^+21]$: Use $\sigma_{\text{min}} \leq \|A(n)\mathbf{v}\|_2 \leq \sigma_{\text{max}}$. $\implies$ lose a factor of $\kappa = \frac{\sigma_{\text{max}}}{\sigma_{\text{min}}}$. 


Better dependence on $\kappa$

- **Idea**: “Guess” the value of $\|A^{(n)}v\|_2$.
- Build an $\epsilon$-net of the line segment $[\sigma_{\text{min}}, \sigma_{\text{max}}]$.
- For any $s$ in the $\epsilon$-net ($s$ is a guess of $\|A^{(n)}v\|_2$), define a truncated sequence $x_{s,1}, \ldots, x_{s,n}$:

$$x_{s,i} := \begin{cases} x_i & \text{if } \|A^{(i)}v\|_2 \leq s, \\ 0 & \text{otherwise}. \end{cases}$$

- Now can prove $|\sum_{i=1}^{n} x_{s,i}| \leq \epsilon \cdot s^2$ by setting $\mu = \epsilon \cdot s^2$.
- Since the size of the $\epsilon$-net is $\propto \kappa$, we only lose another additive $\log(\kappa)$ factor.
Algorithm for adaptive updates

Theorem (Upper bound in adaptive setting). There is a dynamic data structure that maintains an $\epsilon$-approximate LSR solution under adaptive incremental updates, with total time $O(nd \log n + d^5 \text{poly}(\epsilon^{-1}) \log \kappa)$. 
III. Lower Bounds
Theorem (Lower bound). Under the OMv conjecture:

- **High vs low accuracy.** Any dynamic data structure that maintains an \( \epsilon = 1/\text{poly}(n) \)-approximate LSR solution under oblivious incremental updates requires \( \Omega(d^{2-o(1)}) \) amortized cost per iteration.

- **Fully vs partially dynamic.** If the data structure supports adaptive fully dynamic updates, then maintaining 0.01-approximate LSR solution requires \( \Omega(d^{2-o(1)}) \) amortized cost per iteration.
**OMv conjecture.** [HKNS15] In the online matrix vector multiplication (OMv) problem, initially a matrix $M \in \{0, 1\}^{d \times d}$ is given, then a sequence of vectors $v^{(1)}, v^{(2)}, \ldots, v^{(d)} \in \{0, 1\}^d$ are revealed one by one, and the algorithm needs to output $M \cdot v^{(i)}$ in the $i$-th round. The conjecture states that there is no algorithm for OMv with poly($d$) preprocessing time, and $O(d^2 - \epsilon)$ amortized query time.

- Offline: $d^\omega$. Online: $d^3$.
- Only way to speed up matrix vector multiplication is **batching**.
- A unified approach to prove conditional lower bound for dynamic problems.
- Also holds when there are $n = \text{poly}(d)$ queries.
Roadmap

Theorem (Lower bound). Under the OMv conjecture:

- **High vs low accuracy.** Any dynamic data structure that maintains an \( \epsilon = \frac{1}{\text{poly}(n)} \)-approximate LSR solution under *oblivious incremental* updates requires \( \Omega(d^{2-o(1)}) \) amortized cost per iteration. \( \Box \)

- **Fully vs partially dynamic.** If the data structure supports *adaptive fully dynamic* updates, then maintaining 0.01-approximate LSR solution requires \( \Omega(d^{2-o(1)}) \) amortized cost per iteration. \( \Box \)
1/ poly(n)-approximate incremental LSR

OMv conjecture  \[ \xrightarrow{\text{}} \]  O(1/ poly(d))-approx real-valued OMv  \[ \xrightarrow{\text{}} \]  1/ poly(n)-approx incremental LSR ①

O(1/ poly(d))-approx OMv:

- Matrix \( M \in \mathbb{R}^{d \times d} \) has constant eigenvalues.
- Query vectors all have unit norm.
- Allow \( O(1/ \text{poly}(d)) \) additive error in output: \( \| y^{(i)} - M \cdot \nu^{(i)} \|_2 \leq O(1/ \text{poly}(d)) \)

Proof:

- Assume we have a \( 1/(nd^{10}) \)-approx incremental LSR oracle.
- Construct LSR instance: Initially set \((A^{(0)\top}A^{(0)})^{-1} = M\). Add row \( a^{(i)} = \frac{\nu^{(i)}}{nd^5} \).
- Since \( \|a^{(i)}\|_2 \) is small, we always maintain \((A^{(i)\top}A^{(i)})^{-1} \approx M\).
- By Woodbury identity, \( x^{(i)} = x^{(i-1)} + M \cdot a^{(t)} \pm O(\frac{1}{nd^{10}}) \).
- Output \( y^{(i)} = (x^{(i)} - x^{(i-1)}) \cdot nd^5 \) for OMv problem.
0.01-approximate fully dynamic LSR

Proof ideas:

- Assume we have a 0.01-approx fully dynamic LSR oracle.
- Fully dynamic LSR oracle is more powerful:
  - Again add row \( a^{(i)} \propto v^{(i)} \) in \( i \)-th round.
  - Delete the row \( a^{(i)} \) after this round!
- Similar as before, compute output using \( x^{(i)} - x^{(0)} = Mv^{(i)} \pm 0.01 \).
- Need to reduce from a hardness result with constant error.
Hardness amplification

- **Online projection problem:** Initially a projection matrix \(UU^\top \in \mathbb{R}^{d \times d}\) is given, then a sequence of unit vectors \(v^{(1)}, v^{(2)}, \ldots, v^{(d)} \in \mathbb{R}^d\) are revealed one by one. Let \(v^{(i)}_U = UU^\top \cdot v^{(i)}\). The algorithm needs to output:
  - \(O(1/\text{poly}(d))\)-approx solution \(\|y^{(i)} - v^{(i)}_U\|_2 \leq O\left(\frac{1}{\text{poly}(d)}\right)\).
  - \((1/3, 1/\text{poly}(d))\)-approx solution \(\|y^{(i)} - v^{(i)}_U\|_2 \leq \frac{1}{3} \cdot \|v^{(i)}_U\|_2 + O\left(\frac{1}{\text{poly}(d)}\right)\).

- **Hardness amplification:** No \(O(d^{2-\epsilon})\) time algorithm for \(O(1/\text{poly}(d))\)-approx online projection problem. \implies No \(O(d^{2-\epsilon})\) time algorithm for \((1/3, 1/\text{poly}(d))\)-approx online projection problem.

- **Proof:** Given an online projection instance \(UU^\top\) and \(v^{(1)}, \ldots, v^{(n)}\). We have two \(O(1/3, 1/\text{poly}(d))\)-approximate projection oracles:
  - \(\mathbb{P}_U\) that outputs \(y^{(i)}\) s.t. \(\|y^{(i)} - v^{(i)}_U\|_2 \leq \frac{1}{3} \cdot \|v^{(i)}_U\|_2 + O\left(\frac{1}{\text{poly}(d)}\right)\).
  - \(\mathbb{P}_{U_{\perp}}\) that outputs \(w^{(i)}\) s.t. \(\|w^{(i)} - v^{(i)}_{U_{\perp}}\|_2 \leq \frac{1}{3} \cdot \|v^{(i)}_{U_{\perp}}\|_2 + O\left(\frac{1}{\text{poly}(d)}\right)\).
  - **Goal:** Use poly log \(d\) oracle calls to compute \(y^{(i)}\): \(\|y^{(i)} - v^{(i)}_U\|_2 \leq O\left(\frac{1}{\text{poly}(d)}\right)\).
First attempt:

- Call the projection oracle $P_{U_{\perp}}(v)$ to compute $w \approx v_{U_{\perp}}$.
- Remove the component in $U_{\perp}$: compute $v - w$.
- Repeat for $O(\log d)$ times: the component in $U_{\perp}$ is at most $1/\text{poly}(d)$.
- **Problem**: Introduce error in the component in $U$. 
Final algorithm:

- We’ve shown: How to compute \( y \approx v_U \) s.t. \( y \) has nearly zero component in \( U_\perp \).
- Use this algorithm to compute \( w^* \) s.t. its component in \( U \) is nearly zero.
- Again remove the component in \( U_\perp \): compute \( v - w^* \).
- This time we don’t introduce extra error in \( U \).
- Repeat for \( O(\log d) \) times: reduce \( 1/3 \) relative error to \( 1/\text{poly}(d) \) additive error.
Summary and Open problems

- $\epsilon$-approximate dynamic least squares regression

- **Upper bound.** $O(d)$ amortized time when (1) $\epsilon$ is constant, (2) incremental updates, (3) either oblivious or adaptive.

- **Lower bounds.** Under the OMv conjecture:
  - **High vs low accuracy.** If $\epsilon = 1/\text{poly}(n)$, need $\Omega(d^{2-o(1)})$ amortized time.
  - **Fully vs partially dynamic.** If updates are fully dynamic and adaptive, then even constant approximation needs $\Omega(d^{2-o(1)})$ amortized time.

Open problems:

- Improve the $O(d^5)$ term in the total time of adaptive incremental setting?
- Dynamic $\ell_p$ regression?
- Lower bound in fully dynamic and oblivious setting?
- Other reductions from “$(1/3, 1/d^3)$-approximate online projection”?

Thank you!


Rudolph Emil Kalman, *A new approach to linear filtering and prediction problems*. 
