Variable Selection is Hard

Dean P. Foster[1], Howard Karloff, and Justin Thaler[2]

[1]Amazon NYC

[2]Yahoo Labs New York

July 2015

Problem Formulation: (g, h)-Sparse Regression

- Given: An $m \times p$ Boolean matrix B and a positive integer k such that there is a real p-dimensional vector $\mathbf{x}^*$, $\|\mathbf{x}^*\|_0 \leq k$, such that $B\mathbf{x}^* = \mathbf{1}$.
- Goal: Output a p-dimensional vector $\mathbf{x}$ with $\|\mathbf{x}\|_0 \leq k \cdot g(p)$ such that $\|B\mathbf{x} - \mathbf{1}\|^2 \leq h(m, p)$.
- This problem and its noisy variants are central to model design in statistics.
- Sparse solutions are simple, and generalize well.

An Inefficient Algorithm for $(1,0)$-Sparse Regression

- For every k-sparse vector $\mathbf{x}$, check if $B\mathbf{x} = \mathbf{1}$.
- Runs in time $n^{O(k)}$.
- Algorithm does not “cheat” on the sparsity nor the accuracy of the solution.

An Inefficient Algorithm for $(1, 0)$-Sparse Regression

- For every k-sparse vector $\mathbf{x}$, check if $B\mathbf{x} = \mathbf{1}$.
- Runs in time $n^{O(k)}$.
- Algorithm does not "cheat" on the sparsity nor the accuracy of the solution.
- There are many efficient algorithms (e.g. LASSO) that "cheat" only on the accuracy. There are other efficient algorithms that cheat only on the sparsity.
- But all known algorithms may cheat a whole lot if B is ill-conditioned.

An Inefficient Algorithm for $(1, 0)$-Sparse Regression

- For every k-sparse vector $\mathbf{x}$, check if $B\mathbf{x} = \mathbf{1}$.
- Runs in time $n^{O(k)}$.
- Algorithm does not "cheat" on the sparsity nor the accuracy of the solution.
- There are many efficient algorithms (e.g. LASSO) that "cheat" only on the accuracy. There are other efficient algorithms that cheat only on the sparsity.
- But all known algorithms may cheat a whole lot if B is ill-conditioned.
- Main Result of this work: Based on a standard complexity assumption, there is no efficient algorithm that works for general matrices, not even if it is allowed to cheat (a lot) on both the sparsity and accuracy.

Precise Statement of Hardness Result

- **Informal Statement**: There is no efficient algorithm for (g, h)-Sparse Regression, even for if g grows "nearly polynomially quickly" with p, and even if h grows polynomially quickly in p and nearly linearly in m.

Precise Statement of Hardness Result

- **Informal Statement**: There is no efficient algorithm for (g, h)-Sparse Regression, even for if g grows "nearly polynomially quickly" with p, and even if h grows polynomially quickly in p and nearly linearly in m.
- **Formal Statement**: Assume NP $\not\subseteq$ BPTIME$(n^{\text{polylog}(n)})$. Then for any positive constants δ, C_1, C_2, there exist a $g(p)$ in $2^{\Omega(\lg^{1-\delta}(p))}$ and an $h(m, p)$ in $\Omega\left(p^{C_1} \cdot m^{1-C_2}\right)$ such that there is no quasipolynomial-time randomized algorithm for (g, h)-Sparse Regression.

Precise Statement of Hardness Result

- **Informal Statement**: There is no efficient algorithm for (g, h)-Sparse Regression, even for if g grows "nearly polynomially quickly" with p, and even if h grows polynomially quickly in p and nearly linearly in m.
- **Formal Statement**: Assume NP $\not\subseteq$ BPTIME$(n^{\text{polylog}(n)})$. Then for any positive constants δ, C_1, C_2, there exist a $g(p)$ in $2^{\Omega(\lg^{1-\delta}(p))}$ and an $h(m, p)$ in $\Omega\left(p^{C_1} \cdot m^{1-C_2}\right)$ such that there is no quasipolynomial-time randomized algorithm for (g, h)-Sparse Regression.
- Assuming a reasonable conjecture about PCPs, the problem is hard even for some $g(p) \in p^{\Omega(1)}$.

Prior Hardness Results

- Natarajan [1995] and Davis et al. [1997] showed roughly that $(1, 0)$-Sparse Regression is NP-Hard.
  - **“Hardness if algorithm cannot cheat on sparsity or accuracy.”**

Prior Hardness Results

- Natarajan [1995] and Davis et al. [1997] showed roughly that $(1, 0)$-Sparse Regression is NP-Hard.
  - **"Hardness if algorithm cannot cheat on sparsity or accuracy."**
- Arora et al. [1997] and Amaldi and Kahn [1998] showed that there is no polynomial time algorithm for $(2^{\log^{1-\delta}}(p), 1)$-Sparse Regression, assuming that NP $\not\subseteq$ DTIME$(n^{\text{polylog}(n)})$.
  - **"Hardness if algorithm cannot cheat on accuracy."**

Prior Hardness Results

- Natarajan [1995] and Davis et al. [1997] showed roughly that $(1, 0)$-Sparse Regression is NP-Hard.
  - **"Hardness if algorithm cannot cheat on sparsity or accuracy."**
- Arora et al. [1997] and Amaldi and Kahn [1998] showed that there is no polynomial time algorithm for $(2^{\log^{1-\delta}}(p), 1)$-Sparse Regression, assuming that NP $\not\subseteq$ DTIME$(n^{\text{polylog}(n)})$.
  - **"Hardness if algorithm cannot cheat on accuracy."**
- Zhang et al. [2014] showed, roughly, that LASSO's accuracy guarantees in the noisy setting are optimal among all polynomial time algorithms that do not cheat on the sparsity, assuming NP $\not\subseteq$ P$/$poly.
  - **"Hardness if algorithm cannot cheat on sparsity."**

Proof Sketch of Toy Result

- Claim: Any polynomial-time algorithm for $(g(p), 1)$-SPARSE REGRESSION implies an $n^{O(\log\log n)}$-time algorithm for SAT, where $g(p) = (1 - \delta)\ln p$.

Proof Sketch of Toy Result

- Claim: Any polynomial-time algorithm for $(g(p), 1)$-SPARSE REGRESSION implies an $n^{O(\log\log n)}$-time algorithm for SAT, where $g(p) = (1-\delta)\ln p$.
- Proof: Feige gives a reduction from SAT, running in time $n^{O(\log\log n)}$ on SAT instances of size n, to SET COVER, in which the resulting incidence matrix B (whose rows are elements and columns are sets) has the following properties. There is a (known) k such that:
  - If a formula $\phi \in$ SAT, then there is a collection of k disjoint sets which covers the universe, i.e., $B\mathbf{x} = \mathbf{1}$ for some k-sparse $\mathbf{x}$.
  - if $\phi \notin$ SAT, then no collection of at most $k \cdot [(1-\delta)\ln p]$ sets covers the universe. i.e., $B\mathbf{x}$ has at least one entry equal to 0 for any $\|\mathbf{x}\|_0 \leq k \cdot [(1-\delta)\ln p]$. Hence, $\|B\mathbf{x} - \mathbf{1}\|^2 \geq 1$.
  - Any algorithm for $(g(p), 1)$-Sparse regression can distinguish these two cases.