Approximate Nonconvex Optimization and Treewidth

Daniel Bienstock[1]

Triangle Lectures, 2018

[1]IEOR, Columbia University

Collaborators +

- Gonzalo Muñoz: optimizer
- Sebastian Pokutta: optimizer, ML, algorithms, everything
- Mark Zuckergerg, Nuri Ozbay: ex Ph-D students
- Some of us (me) are not ML experts. Some of us are.
- This talk is about theory, theory, theory.
- But we will also outline a possible realistic application of the methodology.

An “application”

- Deep Learning is receiving significant attention due to its impressive performance.

An “application”

- Deep Learning is receiving significant attention due to its impressive performance.
- It has been only recently that results regarding the complexity of training deep neural networks have been obtained.

An “application”

- Deep Learning is receiving significant attention due to its impressive performance.
- It has been only recently that results regarding the complexity of training deep neural networks have been obtained.
- We will show that large classes of Neural Networks can be trained to proved near optimality using linear programs whose size is linear on the training data.
- SGD has running time linear on the training data, but it does not offer optimality guarantees.

Empirical Risk Minimization problem

Empirical Risk Minimization problem

Given:

- D data points $(\hat{x}_i, \hat{y}_i)$, $i = 1, \ldots, D$
- $\hat{x}_i \in \mathbb{R}^n,\ \hat{y}_i \in \mathbb{R}^m$
- A loss function $\ell : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$ (not necessarily convex)

Empirical Risk Minimization problem

Given:

- D data points $(\hat{x}_i, \hat{y}_i)$, $i = 1, \ldots, D$
- $\hat{x}_i \in \mathbb{R}^n$, $\hat{y}_i \in \mathbb{R}^m$
- A loss function $\ell : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$ (not necessarily convex)

Compute $f : \mathbb{R}^n \to \mathbb{R}^m$ to solve

$$
\begin{aligned}
&\min_f \frac{1}{D} \sum_{i=1}^{D} \ell(f(\hat{x}^i), \hat{y}^i) && \text{(+ optional regularizer } \Phi(f)\text{)} \\
&f \in F && \text{(some class)}
\end{aligned}
$$

Function parameterization

$$\min_f \frac{1}{D} \sum_{i=1}^{D} \ell(f(\hat{x}^i), \hat{y}^i) \qquad \text{(+ optional regularizer } \Phi(f))$$

$$f \in F \quad \text{(some class)}$$

We assume family F (statisticians' hypothesis) is parameterized: there exists f such that

$$F = \{f(x, \theta) \ : \ \theta \in \Theta \subseteq [-1, 1]^N\}.$$

Function parameterization

$$\min_f \frac{1}{D} \sum_{i=1}^{D} \ell(f(\hat{x}^i), \hat{y}^i) \qquad \text{(+ optional regularizer } \Phi(f))$$

$$f \in F \quad \text{(some class)}$$

We assume family F (statisticians' hypothesis) is parameterized: there exists f such that

$$F = \{f(x, \theta) \ : \ \theta \in \Theta \subseteq [-1, 1]^N\}.$$

Using this notation the ERM problem becomes

$$\min_{\theta \in \Theta} \frac{1}{D} \sum_{i=1}^{D} \ell(f(\hat{x}^i, \theta), \hat{y}^i)$$

Empirical Risk Minimization problem

$$\min_f \frac{1}{D} \sum_{i=1}^{D} \ell(f(\hat{x}^i), \hat{y}^i) \qquad \text{(+ optional regularizer } \Phi(f)\text{)}$$

$$f \in F \quad \text{(some class)}$$

Examples:

- Linear Regression. $f(x) \;=\; Ax + b$ with ℓ_2-loss.
- Binary Classification. Varying f families and cross-entropy loss: $\ell(p, y) = -y \log(p) - (1 - y) \log(1 - p)$
- Neural Networks with k layers. $f(x) \;=\; T_{k+1} \circ \sigma \circ T_k \circ \sigma \ldots \circ \sigma \circ T_1(x)$, each T_j affine.

Deep Networks (with 2-norm squared loss)

- D data points $(\hat{x}_i, \hat{y}_i),\ 1 \leq i \leq D$, $\hat{x}_i \in \mathbb{R}^n$, $\hat{y}_i \in \mathbb{R}^m$

Deep Networks (with 2-norm squared loss)

- D data points $(\hat{x}_i, \hat{y}_i)$, $1 \leq i \leq D$, $\hat{x}_i \in \mathbb{R}^n$, $\hat{y}_i \in \mathbb{R}^m$
- Compute $f : \mathbb{R}^n \to \mathbb{R}^m$ to minimize

$$\frac{1}{D} \sum_{i=1}^{D} \left\| \hat{y}_i - f(\hat{x}_i) \right\|^2$$

Deep Networks (with 2-norm squared loss)

- D data points $(\hat{x}_i, \hat{y}_i)$, $1 \le i \le D$, $\hat{x}_i \in \mathbb{R}^n$, $\hat{y}_i \in \mathbb{R}^m$
- Compute $f : \mathbb{R}^n \to \mathbb{R}^m$ to minimize

$$\frac{1}{D} \sum_{i=1}^{D} \|\hat{y}_i - f(\hat{x}_i)\|^2$$

- $f = T_{k+1} \circ \sigma \circ T_k \circ \sigma \ldots \circ \sigma \circ T_1$

Deep Networks (with 2-norm squared loss)

- D data points $(\hat{x}_i, \hat{y}_i)$, $1 \leq i \leq D$, $\hat{x}_i \in \mathbb{R}^n$, $\hat{y}_i \in \mathbb{R}^m$
- Compute $f : \mathbb{R}^n \to \mathbb{R}^m$ to minimize

$$\frac{1}{D} \sum_{i=1}^{D} \|\hat{y}_i - f(\hat{x}_i)\|^2$$

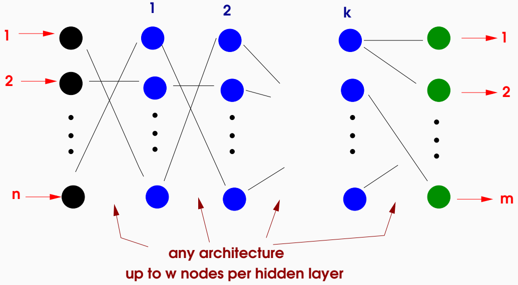
- $f = T_{k+1} \circ \sigma \circ T_k \circ \sigma \ldots \circ \sigma \circ T_1$
- Each T_i affine: $T_i(y) = A_i y + b_i$. Example $\sigma(t) = \max\{0, t\}$ (ReLu)

Deep Networks (with 2-norm squared loss)

- D data points $(\hat{x}_i, \hat{y}_i)$, $1 \leq i \leq D$, $\hat{x}_i \in \mathbb{R}^n$, $\hat{y}_i \in \mathbb{R}^m$
- Compute $f : \mathbb{R}^n \to \mathbb{R}^m$ to minimize

$$\frac{1}{D} \sum_{i=1}^{D} \|\hat{y}_i - f(\hat{x}_i)\|^2$$

- $f = T_{k+1} \circ \sigma \circ T_k \circ \sigma \ldots \circ \sigma \circ T_1$
- Each T_i affine: $T_i(y) = A_i y + b_i$. Example $\sigma(t) = \max\{0, t\}$ (ReLu)
- A_1 is $n \times w$, A_{k+1} is $w \times m$, A_i is $w \times w$ otherwise.



What we know for Neural Nets

Hardness Result

Problem: Compute $f : \mathbb{R}^n \to \mathbb{R}^m$ to minimize

$$\frac{1}{D}\sum_{i=1}^{D} \|\hat{y}_i - f(\hat{x}_i)\|^2$$

Hardness Result

Problem: Compute $f : \mathbb{R}^n \to \mathbb{R}^m$ to minimize

$$\frac{1}{D}\sum_{i=1}^{D} \left\|\hat{y}_i - f(\hat{x}_i)\right\|^2$$

where $f = T_{k+1} \circ \sigma \circ T_k \circ \sigma \ldots \circ \sigma \circ T_1$

Hardness Result

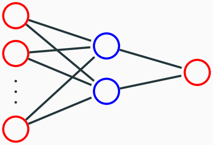
Problem: Compute $f : \mathbb{R}^n \to \mathbb{R}^m$ to minimize

$$\frac{1}{D}\sum_{i=1}^{D} \|\hat{y}_i - f(\hat{x}_i)\|^2$$

where $f = T_{k+1} \circ \sigma \circ T_k \circ \sigma \ldots \circ \sigma \circ T_1$

Theorem (Blum and Rivest 1992)

When $\ell \in$ (absolute value, 2-norm squared) training is NP-hard even if $k = 1$ (only 3 nodes), $D \in O(n)$, $m = 1$, $\hat{x}^i \in \{0,1\}^n$, $\hat{y}^i \in \{0,1\}$ and weights are ± 1.



Exact Training Complexity

Problem: Compute $f : \mathbb{R}^n \to \mathbb{R}^m$ to minimize

$$\frac{1}{D}\sum_{i=1}^{D} \|\hat{y}_i - f(\hat{x}_i)\|^2$$

Exact Training Complexity

Problem: Compute $f : \mathbb{R}^n \to \mathbb{R}^m$ to minimize

$$\frac{1}{D}\sum_{i=1}^{D} \|\hat{y}_i - f(\hat{x}_i)\|^2$$

where $f = T_{k+1} \circ \sigma \circ T_k \circ \sigma \ldots \circ \sigma \circ T_1$

Exact Training Complexity

Problem: Compute $f : \mathbb{R}^n \to \mathbb{R}^m$ to minimize

$$\frac{1}{D}\sum_{i=1}^{D} \|\hat{y}_i - f(\hat{x}_i)\|^2$$

where $f = T_{k+1} \circ \sigma \circ T_k \circ \sigma \ldots \circ \sigma \circ T_1$

Theorem (Goel Kanade Klivans Thaler 2016)

(Abridged!) There is an algorithm for improper *learning of networks of ReLUs*

Exact Training Complexity

Problem: Compute $f : \mathbb{R}^n \to \mathbb{R}^m$ to minimize

$$\frac{1}{D}\sum_{i=1}^{D} \|\hat{y}_i - f(\hat{x}_i)\|^2$$

where $f = T_{k+1} \circ \sigma \circ T_k \circ \sigma \ldots \circ \sigma \circ T_1$

Theorem (Goel Kanade Klivans Thaler 2016)

(Abridged!) There is an algorithm for improper *learning of networks of ReLUs*

- *Running time is (more than) doubly exponential in e.g. number of layers.*
- *For two layers, exponential in* $\epsilon^{-\Theta(1)}$ *and polynomial in* D

Exact Training Complexity

Problem: Compute $f : \mathbb{R}^n \to \mathbb{R}^m$ to minimize

$$\frac{1}{D} \sum_{i=1}^{D} \|\hat{y}_i - f(\hat{x}_i)\|^2$$

Exact Training Complexity

Problem: Compute $f : \mathbb{R}^n \to \mathbb{R}^m$ to minimize

$$\frac{1}{D} \sum_{i=1}^{D} \|\hat{y}_i - f(\hat{x}_i)\|^2$$

where $f = T_{k+1} \circ \sigma \circ T_k \circ \sigma \ldots \circ \sigma \circ T_1$

Exact Training Complexity

Problem: Compute $f : \mathbb{R}^n \to \mathbb{R}^m$ to minimize

$$\frac{1}{D} \sum_{i=1}^{D} \|\hat{y}_i - f(\hat{x}_i)\|^2$$

where $f = T_{k+1} \circ \sigma \circ T_k \circ \sigma \ldots \circ \sigma \circ T_1$

Theorem (Arora Basu Mianjy Mukherjee 2018)

If $k = 1$ (one "hidden layer") there is an exact training algorithm of complexity

Exact Training Complexity

Problem: Compute $f : \mathbb{R}^n \to \mathbb{R}^m$ to minimize

$$\frac{1}{D}\sum_{i=1}^{D} \|\hat{y}_i - f(\hat{x}_i)\|^2$$

where $f = T_{k+1} \circ \sigma \circ T_k \circ \sigma \ldots \circ \sigma \circ T_1$

Theorem (Arora Basu Mianjy Mukherjee 2018)

If $k = 1$ (one "hidden layer") there is an exact training algorithm of complexity

$$O\left(2^w \boldsymbol{D}^{\boldsymbol{nw}} poly(D, n, w)\right)$$

*Polynomial in the size **D** of the data set, for fixed n, w.*

$\boldsymbol{n} =$ *dimension of input vectors,*

Exact Training Complexity

Problem: Compute $f : \mathbb{R}^n \to \mathbb{R}^m$ to minimize

$$\frac{1}{D}\sum_{i=1}^{D} \|\hat{y}_i - f(\hat{x}_i)\|^2$$

where $f = T_{k+1} \circ \sigma \circ T_k \circ \sigma \ldots \circ \sigma \circ T_1$

Theorem (Arora Basu Mianjy Mukherjee 2018)

If $k = 1$ (one "hidden layer") there is an exact training algorithm of complexity

$$O\left(2^w \mathbf{D}^{\mathbf{nw}} poly(D, n, w) \right)$$

*Polynomial in the size **D** of the data set, for fixed n, w.*

$\mathbf{n} =$ *dimension of input vectors,* $\mathbf{w} =$ *size of internal layers*

Theorem

For every $\epsilon > 0$, ℓ, $\Theta \subseteq [-1, 1]^N$ and D, there is a polytope with variables (θ, x^i, y^i, L_i) of size

$$O\left((2\mathcal{L}/\epsilon)^{N+n+m+1}\, D\right)$$

($\boldsymbol{\mathcal{L}}$ = *largest Lipshitz constant of any* $\boldsymbol{\sigma}$, *and* $\mathbf{N = O(wk)}$)

Theorem

For every $\epsilon > 0$, ℓ, $\Theta \subseteq [-1,1]^N$ and D, there is a polytope with variables (θ, x^i, y^i, L_i) of size

$$O\left((2\mathcal{L}/\epsilon)^{N+n+m+1}\, D\right)$$

($\boldsymbol{\mathcal{L}}$ = largest Lipshitz constant of any $\boldsymbol{\sigma}$, and $\mathbf{N = O(wk)}$)

s.t. $\forall$ data set $(\hat{X}, \hat{Y}) = (\hat{x}^i, \hat{y}^i)$, $i = 1, \ldots D$, there is a face $\mathcal{F}_{\hat{X},\hat{Y}}$ with

$$\min_{\theta \in \Theta} \frac{1}{D} \sum_{i=1}^{D} L_i$$

$$(\theta, L) \in \mathrm{proj}(\mathcal{F}_{\hat{X},\hat{Y}})$$

provides an ϵ-approximation to ERM with data $\hat{X}, \hat{Y}$.

A uniform ("universal") linear program of size linear in the quantity of training data.

Our main toolset

Treewidth

Treewidth is a parameter that measures how *tree-like* a graph is.

Treewidth

Treewidth is a parameter that measures how *tree-like* a graph is.

Definition

The treewidth of graph G, is the smallest clique number of any chordal supergraph of G, minus 1.

Treewidth

Treewidth is a parameter that measures how *tree-like* a graph is.

Definition

The treewidth of graph G, is the smallest clique number of any chordal supergraph of G, minus 1.

- **OLD** concept, but term coined by Robertson and Seymour (1980s).
- Informal definition: graphs with small treewidth are the "simple" graphs
- **Many** equivalent definitions.
- Trees have treewidth 1
- Cycles have treewidth 2
- K_n has treewidth $n - 1$
- the $k \times k$ planar grid has treewidth k

Treewidth

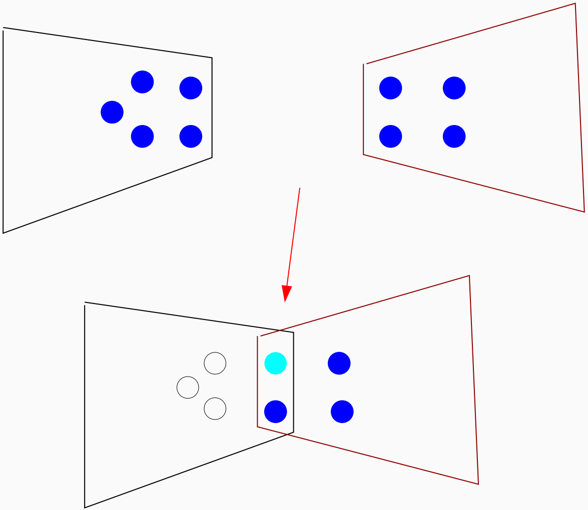
Informal algorithmic definition:

Treewidth

Informal algorithmic definition:

- Start with with m “bags” with up to $\omega + 1$ vertices each, numbered $1, 2, \ldots, m$. “Bag” = graph.
- Each bag includes some edges between vertices in the bag.
- Start of procedure: each bag is a “processed unit”. All vertices are “boundary” vertices.
- Inductive step: take two processed units. Idenfify (“glue”) some of the boundary with the same number of vertices in one unit with same number of boundary vertices of the other, forming a new processing unit.
- The boundary of the new unit will be a subset of the union of the two boundaries, of size $\leq \omega + 1$

Treewidth



Treewidth: literature review

Treewidth is a parameter that measures how *tree-like* a graph is.

Treewidth: literature review

Treewidth is a parameter that measures how *tree-like* a graph is.

Definition

The treewidth of graph G, is the smallest clique number of any chordal supergraph of G, minus 1.

Treewidth: literature review

Treewidth is a parameter that measures how *tree-like* a graph is.

Definition

The treewidth of graph G, is the smallest clique number of any chordal supergraph of G, minus 1.

- Intersection graph of subtrees of a tree, Robertson and Seymour (1980s).
  - Core concept in proof of Wagner Conjecture, 1984-2002.
  - Early result: given a planar graph H any graph with no H minor has tree-width at most $f(H)$.

Treewidth: literature review

Treewidth is a parameter that measures how *tree-like* a graph is.

Definition

The treewidth of graph G, is the smallest clique number of any chordal supergraph of G, minus 1.

- Intersection graph of subtrees of a tree, Robertson and Seymour (1980s).
  - Core concept in proof of Wagner Conjecture, 1984-2002.
  - Early result: given a planar graph H any graph with no H minor has tree-width at most $f(H)$.
- “K-trees”, perfect graph literature, (1970s and after)

Treewidth: literature review

Treewidth is a parameter that measures how *tree-like* a graph is.

Definition

The treewidth of graph G, is the smallest clique number of any chordal supergraph of G, minus 1.

- Intersection graph of subtrees of a tree, Robertson and Seymour (1980s).
  - Core concept in proof of Wagner Conjecture, 1984-2002.
  - Early result: given a planar graph H any graph with no H minor has tree-width at most $\boldsymbol{f(H)}$.
- “K-trees”, perfect graph literature, (1970s and after)
- Junction tree theorem, Lauritzen (1996), Pearle (1982). Graphical Models, Belief Propagation.

Treewidth: literature review

Treewidth is a parameter that measures how *tree-like* a graph is.

Definition

The treewidth of graph G, is the smallest clique number of any chordal supergraph of G, minus 1.

- Intersection graph of subtrees of a tree, Robertson and Seymour (1980s).
  - Core concept in proof of Wagner Conjecture, 1984-2002.
  - Early result: given a planar graph H any graph with no H minor has tree-width at most $f(H)$.
- "K-trees", perfect graph literature, (1970s and after)
- Junction tree theorem, Lauritzen (1996), Pearle (1982). Graphical Models, Belief Propagation.
- Nonserial Dynamic Programming, Nemhauser (1964). Bertele and Brioschi (1972).

Second tool: approximation through digitization

Let $0 \leq z \leq 1$ and let $L > 0$ integral
Then we can approximate

$$z \quad \approx \quad \textstyle\sum_{k=1}^{L} 2^{-k}\, y_k, \quad \text{where each } y_k \text{ binary.}$$

Second tool: approximation through digitization

Let $0 \leq z \leq 1$ and let $L > 0$ integral
Then we can approximate

$$z \quad \approx \quad \sum_{k=1}^{L} 2^{-k} y_k, \quad \text{where each } y_k \text{ binary.} \quad \text{Error} \leq 2^{-L}$$

Let $0 \leq z \leq 1$ and let $L > 0$ integral
Then we can approximate

$$z \quad \approx \quad \sum_{k=1}^{L} 2^{-k}\, y_k, \quad \text{where each } y_k \text{ binary.} \qquad \text{Error} \leq 2^{-L}$$

- Suppose $F : \mathbb{R}^m \rightarrow \mathbb{R}$ with Lipschitz constant $\mathcal{L}$

Second tool: approximation through digitization

Let $0 \leq z \leq 1$ and let $L > 0$ integral
Then we can approximate

$$z \quad \approx \quad \sum_{k=1}^{L} 2^{-k} y_k, \quad \text{where each } y_k \text{ binary.} \quad \text{Error} \leq 2^{-L}$$

- Suppose $F : \mathbb{R}^m \to \mathbb{R}$ with Lipschitz constant $\mathcal{L}$
- Digitize (approximate) each component of $x \in \mathbb{R}^m$

Second tool: approximation through digitization

Let $0 \leq z \leq 1$ and let $L > 0$ integral
Then we can approximate

$$z \quad \approx \quad \sum_{k=1}^{L} 2^{-k} y_k, \quad \text{where each } y_k \text{ binary.} \quad \text{Error} \leq 2^{-L}$$

- Suppose $F : \mathbb{R}^m \rightarrow \mathbb{R}$ with Lipschitz constant $\mathcal{L}$
- Digitize (approximate) each component of $x \in \mathbb{R}^m$
- So $x \rightarrow y$ (the binary approximation). $y \in \mathbb{R}^{mL}$

Second tool: approximation through digitization

Let $0 \leq z \leq 1$ and let $L > 0$ integral
Then we can approximate

$$z \quad \approx \quad \sum_{k=1}^{L} 2^{-k} y_k, \quad \text{where each } y_k \text{ binary.} \quad \text{Error} \leq 2^{-L}$$

- Suppose $F : \mathbb{R}^m \rightarrow \mathbb{R}$ with Lipschitz constant $\mathcal{L}$
- Digitize (approximate) each component of $x \in \mathbb{R}^m$
- So $x \rightarrow y$ (the binary approximation). $y \in \mathbb{R}^{mL}$
- And $F(x)$ replaced by $\hat{F}(y)$

Second tool: approximation through digitization

Let $0 \leq z \leq 1$ and let $L > 0$ integral
Then we can approximate

$$z \quad \approx \quad \sum_{k=1}^{L} 2^{-k} y_k, \quad \text{where each } y_k \text{ binary.} \qquad \text{Error} \leq 2^{-L}$$

- Suppose $F : \mathbb{R}^m \rightarrow \mathbb{R}$ with Lipschitz constant $\mathcal{L}$
- Digitize (approximate) each component of $x \in \mathbb{R}^m$
- So $x \rightarrow y$ (the binary approximation). $y \in \mathbb{R}^{mL}$
- And $F(x)$ replaced by $\hat{F}(y)$
- $\hat{F}(y) \leq F(x) + m\mathcal{L}\, 2^{-L}$

Final ingredient: lifted formulations

Lovász and Schrijver, Sherali and Adams, ~ 1990

- $0/1$ integer programming:

Final ingredient: lifted formulations

Lovász and Schrijver, Sherali and Adams, $\sim$ 1990

- **0/1** integer programming: $\{x \in \{0,1\}^n \,:\, Ax \leq b\}$
- Key idea: **add** new variables that express conjunctions:

Final ingredient: lifted formulations

Lovász and Schrijver, Sherali and Adams, $\sim$ 1990

- 0/1 integer programming: $\{x \in \{0,1\}^n \ : \ Ax \leq b\}$
- Key idea: add new variables that express conjunctions:
    - $x[\{1,2\},\{3\}]$ which equals 1 iff $x_1 = x_2 = 1$ and $x_3 = 0$

Final ingredient: lifted formulations

Lovász and Schrijver, Sherali and Adams, $\sim$ 1990

- $0/1$ integer programming: $\{x \in \{0,1\}^n \ : \ Ax \leq b\}$
- Key idea: **add** new variables that express conjunctions:
  - $x[\{1,2\},\{3\}]$ which equals 1 iff $x_1 = x_2 = 1$ and $x_3 = 0$
  - Same as monomial $x_1 x_2 (1 - x_3)$

Final ingredient: lifted formulations

Lovász and Schrijver, Sherali and Adams, $\sim$ 1990

- $0/1$ integer programming: $\{x \in \{0,1\}^n \; : \; Ax \leq b\}$
- Key idea: add new variables that express conjunctions:
  - $x[\{1,2\},\{3\}]$ which equals 1 iff $x_1 = x_2 = 1$ and $x_3 = 0$
  - Same as monomial $x_1 x_2 (1 - x_3)$
  - can strengthen the formulation, e.g. if
    $2x_1 + x_3 + x_4 \leq 1$ is a constraint of $Ax \leq b$
    then can say: $2x[\{1,2\},\{3\}] + x[\{1,2,4\},\{3\}] \leq x[\{1,2\},\{3\}]$
  - And more, e.g. $x[\{1,2\},\{3\}] + x[\{1\},\{2,3\}] = x[\{1\},\{3\}]$
- "Level-k" formulation: only use monomials with up to k terms

Theorem: (2004)

Consider $\{x \in \{0,1\}^n \; : \; Ax \leq b\}$ where $A \geq 0$

Final ingredient: lifted formulations

Lovász and Schrijver, Sherali and Adams, $\sim$ 1990

- 0/1 integer programming: $\{x \in \{0,1\}^n \ : \ Ax \leq b\}$
- Key idea: add new variables that express conjunctions:
  - $x[\{1,2\},\{3\}]$ which equals 1 iff $x_1 = x_2 = 1$ and $x_3 = 0$
  - Same as monomial $x_1 x_2 (1 - x_3)$
  - can strengthen the formulation, e.g. if $2x_1 + x_3 + x_4 \leq 1$ is a constraint of $Ax \leq b$ then can say: $2x[\{1,2\},\{3\}] + x[\{1,2,4\},\{3\}] \leq x[\{1,2\},\{3\}]$
  - And more, e.g. $x[\{1,2\},\{3\}] + x[\{1\},\{2,3\}] = x[\{1\},\{3\}]$
- "Level-k" formulation: only use monomials with up to k terms

Theorem: (2004)
Consider $\{x \in \{0,1\}^n \ : \ Ax \leq b\}$ where $A \geq 0$ (a packing problem).

Final ingredient: lifted formulations

Lovász and Schrijver, Sherali and Adams, $\sim$ 1990

- 0/1 integer programming: $\{x \in \{0,1\}^n \ : \ Ax \leq b\}$
- Key idea: add new variables that express conjunctions:
  - $x[\{1,2\},\{3\}]$ which equals 1 iff $x_1 = x_2 = 1$ and $x_3 = 0$
  - Same as monomial $x_1 x_2 (1 - x_3)$
  - can strengthen the formulation, e.g. if
    $2x_1 + x_3 + x_4 \leq 1$ is a constraint of $Ax \leq b$
    then can say: $2x[\{1,2\},\{3\}] + x[\{1,2,4\},\{3\}] \leq x[\{1,2\},\{3\}]$
  - And more, e.g. $x[\{1,2\},\{3\}] + x[\{1\},\{2,3\}] = x[\{1\},\{3\}]$
- "Level-k" formulation: only use monomials with up to k terms

Theorem: (2004)
Consider $\{x \in \{0,1\}^n \ : \ Ax \leq b\}$ where $A \geq 0$ (a packing problem).
then "level-k" formulation implies all valid inequalities $c^T x \leq d$

Final ingredient: lifted formulations

Lovász and Schrijver, Sherali and Adams, $\sim$ 1990

- $0/1$ integer programming: $\{x \in \{0,1\}^n \ : \ Ax \leq b\}$
- Key idea: add new variables that express conjunctions:
  - $x[\{1,2\},\{3\}]$ which equals 1 iff $x_1 = x_2 = 1$ and $x_3 = 0$
  - Same as monomial $x_1 x_2 (1 - x_3)$
  - can strengthen the formulation, e.g. if
    $2x_1 + x_3 + x_4 \leq 1$ is a constraint of $Ax \leq b$
    then can say: $2x[\{1,2\},\{3\}] + x[\{1,2,4\},\{3\}] \leq x[\{1,2\},\{3\}]$
  - And more, e.g. $x[\{1,2\},\{3\}] + x[\{1\},\{2,3\}] = x[\{1\},\{3\}]$
- "Level-k" formulation: only use monomials with up to k terms

Theorem: (2004)
Consider $\{x \in \{0,1\}^n \ : \ Ax \leq b\}$ where $A \geq 0$ (a packing problem).
then "level-k" formulation implies all valid inequalities $c^T x \leq d$
where the subgraph of the intersection graph of A,

Final ingredient: lifted formulations

Lovász and Schrijver, Sherali and Adams, $\sim$ 1990

- $0/1$ integer programming: $\{x \in \{0,1\}^n \ : \ Ax \leq b\}$
- Key idea: **add** new variables that express conjunctions:
  - $x[\{1,2\},\{3\}]$ which equals 1 iff $x_1 = x_2 = 1$ and $x_3 = 0$
  - Same as monomial $x_1 x_2 (1 - x_3)$
  - can strengthen the formulation, e.g. if
    $2x_1 + x_3 + x_4 \leq 1$ is a constraint of $Ax \leq b$
    then can say: $2x[\{1,2\},\{3\}] + x[\{1,2,4\},\{3\}] \leq x[\{1,2\},\{3\}]$
  - And more, e.g. $x[\{1,2\},\{3\}] + x[\{1\},\{2,3\}] = x[\{1\},\{3\}]$
- "Level-k" formulation: only use monomials with up to k terms

Theorem: (2004)
Consider $\{x \in \{0,1\}^n \ : \ Ax \leq b\}$ where $A \geq 0$ (a packing problem).
then "level-k" formulation implies all valid inequalities $c^T x \leq d$
where the subgraph of the intersection graph of A, induced by c

Final ingredient: lifted formulations

Lovász and Schrijver, Sherali and Adams, $\sim$ 1990

- $0/1$ integer programming: $\{x \in \{0,1\}^n \ : \ Ax \leq b\}$
- Key idea: **add** new variables that express conjunctions:
  - $x[\{1,2\},\{3\}]$ which equals 1 iff $x_1 = x_2 = 1$ and $x_3 = 0$
  - Same as monomial $x_1 x_2 (1 - x_3)$
  - can strengthen the formulation, e.g. if
    $2x_1 + x_3 + x_4 \leq 1$ is a constraint of $Ax \leq b$
    then can say: $2x[\{1,2\},\{3\}] + x[\{1,2,4\},\{3\}] \leq x[\{1,2\},\{3\}]$
  - And more, e.g. $x[\{1,2\},\{3\}] + x[\{1\},\{2,3\}] = x[\{1\},\{3\}]$
- "Level-k" formulation: only use monomials with up to k terms

Theorem: (2004)
Consider $\{x \in \{0,1\}^n \ : \ Ax \leq b\}$ where $A \geq 0$ (a packing problem).
then "level-k" formulation implies all valid inequalities $c^T x \leq d$
where the subgraph of the intersection graph of A, induced by c
has **treewidth** $\leq k$

Approximate optimization of well-behaved functions

Prototype problem:

$$\begin{aligned} c^* \ \doteq \ \min\ & c^T x \\ \text{s.t. } & f_i(x) \leq 0, \qquad i = 1, \ldots, m \\ & x \in [0,1]^n \end{aligned}$$

An extension of work in Bienstock and Muñoz (2015).

Theorem

Suppose the intersection graph has tree-width ω *and let* $\mathcal{L} = \max_i \mathcal{L}_i$.

Approximate optimization of well-behaved functions

Prototype problem:

$$
\begin{aligned}
c^* \;\doteq\; \min\; & c^T x \\
\text{s.t. } & f_i(x) \le 0, \qquad i = 1, \ldots, m \\
& x \in [0,1]^n
\end{aligned}
$$

An extension of work in Bienstock and Muñoz (2015).

Theorem

Suppose the intersection graph has tree-width ω and let $\mathcal{L} = \max_i \mathcal{L}_i$. Then, for every $\epsilon > 0$ there is an LP relaxation of size

$$O\left((\mathcal{L}/\epsilon)^{\omega+1}\, n \right)$$

that guarantees ϵ-optimality and -feasibility errors.

Approximate optimization of well-behaved functions

Prototype problem:

$$
\begin{aligned}
c^* \ \doteq \ \min\ & c^T x \\
\text{s.t. } & f_i(x) \le 0, \qquad i = 1, \ldots, m \\
& x \in [0,1]^n
\end{aligned}
$$

An extension of work in Bienstock and Muñoz (2015).

Theorem

Suppose the intersection graph has tree-width ω and let $\mathcal{L} = \max_i \mathcal{L}_i$. Then, for every $\epsilon > 0$ there is an LP relaxation of size

$$O\left((\mathcal{L}/\epsilon)^{\omega+1}\, n\right)$$

that guarantees ϵ-optimality and -feasibility errors.

$$c^T\hat{x} \ \le \ c^* + O(\epsilon), \quad f_i(\hat{x}) \ \le \ O(\epsilon)\ \forall i$$

Application to ERM problem

We now apply the LP approximation result to:

$$\min_{\theta\in\Theta} \frac{1}{D}\sum_{i=1}^{D} \ell(f(\hat{x}^i,\theta),\hat{y}^i) \quad 1\leq i\leq D$$

with $\Theta \subseteq [-1,1]^N$, $\hat{x}^i \in [-1,1]^n$ and $\hat{y}^i \in [-1,1]^m$.

Application to ERM problem

We now apply the LP approximation result to:

$$\min_{\theta\in\Theta} \frac{1}{D}\sum_{i=1}^{D} \ell(f(\hat{x}^i,\theta),\hat{y}^i) \quad 1 \leq i \leq D$$

with $\Theta \subseteq [-1,1]^N$, $\hat{x}^i \in [-1,1]^n$ and $\hat{y}^i \in [-1,1]^m$. We use the epigraph formulation:

$$\min_{\theta\in\Theta} \frac{1}{D}\sum_{i=1}^{D} L_i$$

$$L_i \geq \ell(f(\hat{x}^i,\theta),\hat{y}^i) \quad 1 \leq i \leq D$$

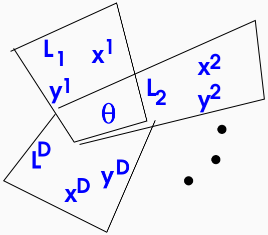
Let $\mathcal{L}$ be the Lipschitz constant for $g(x,y,\theta) \doteq \ell(f(x,\theta),y)$ over $[-1,1]^{n+m+N}$.

Proof Sketch

Every system of constraints of the type

$$L_i \geq \ell(f(x^i, \theta), y^i) \quad 1 \leq i \leq D$$

has an intersection graph with the following structure:



resulting in a formulation with *treewidth* at most $N + n + m + 1$

LP size details

Thus the LP size given by the expression

$$O\left((\mathcal{L}/\epsilon)^{\omega+1}\, n\right)$$

becomes

$$O\left((2\mathcal{L}/\epsilon)^{N+n+m+1}\, D\right)$$

LP size details

Thus the LP size given by the expression

$$O\left((\mathcal{L}/\epsilon)^{\omega+1}\, n\right)$$

becomes

$$O\left((2\mathcal{L}/\epsilon)^{N+n+m+1}\, D\right)$$

The key to linear dependence on D lies in the fact that the D does not add to the treewidth.

LP size details

Thus the LP size given by the expression

$$O\left((\mathcal{L}/\epsilon)^{\omega+1}\, n\right)$$

becomes

$$O\left((2\mathcal{L}/\epsilon)^{N+n+m+1}\, D\right)$$

The key to linear dependence on D lies in the fact that the D does not add to the treewidth.

Different architectures $\rightarrow$ N and $\mathcal{L}$.

Architecture-Specific Consequences

Theorem (Arora, Basu, Mianjy and Mukherjee 2018)

If $k = 1$ *(one "hidden layer") and* $m = 1$ *there is an exact training algorithm of complexity*

Theorem (Arora, Basu, Mianjy and Mukherjee 2018)

If $k = 1$ (one "hidden layer") and $m = 1$ there is an exact training algorithm of complexity

$$O\left(\, 2^{w} D^{nw} poly(D, n, w) \,\right)$$

Polynomial in the size of the data set, for fixed n, w.

Consequence of our result

Consequence of our result

If the entries of A_i, b_i are required to be in $[-1, 1]$, for any k, n, m, w, ϵ there is a uniform LP of size

$$O\left((nw/\epsilon)^{k(n+m+N+1)}\, D\right)$$

with the same guarantees as before.

Consequence of our result

If the entries of A_i, b_i are required to be in $[-1, 1]$, for any k, n, m, w, ϵ there is a uniform LP of size

$$O\left((nw/\epsilon)^{k(n+m+N+1)}\, D\right)$$

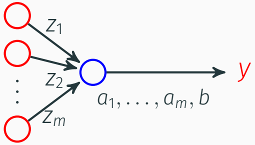
with the same guarantees as before.

Core of the proof: In a DNN with k hidden layers the Lipschitz constant of $g(x, y, \theta)$ over $[-1, 1]^{n+m+N}$ is $\sim nw^k$.

The Blum-Rivest setup (Binarized Neural Networks)

The Blum-Rivest setup (Binarized Neural Networks)

Activation units:

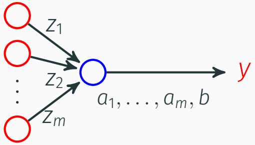


With $z \in \{0,1\}^m$,

$$y = \begin{cases} 1, & \text{if } a^T z > b \\ 0, & \text{otherwise.} \end{cases}$$

The Blum-Rivest setup (Binarized Neural Networks)

Activation units:



With $z \in \{0, 1\}^m$,

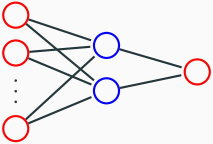
$$y = \begin{cases} 1, & \text{if } a^T z > b \\ 0, & \text{otherwise.} \end{cases}$$

Network with n binary inputs, m binary outputs, k layers

The Blum-Rivest setup (Binarized Neural Networks)

Theorem (Blum and Rivest 1992)

When $\ell \in$ (absolute value, 2-norm squared) training is NP-hard even if $k = 1$ (only 3 nodes), $D \in O(n)$, $m = 1$ and weights are ± 1.



Theorem

Consider a BNN with $m = 1$ and ℓ arbitrary. There is a uniform LP of size

$$O(2^{poly(k,n,w)}D)$$

that solves ERM exactly for any input data.

Concluding comments

- The results can be improved by considering the sparsity of the network itself.
- Training using this approach *generalizes*. Meaning, using *enough*[1] data points we get an approximation to the "true" Risk Minimization problem.

[1] depends on $\mathcal{L}$ and ϵ

Thank you!

Did we say something about practicality?

We are solving Linear Programs. There are classical examples of linear programs that are solved **incrementally**.

Did we say something about practicality?

We are solving Linear Programs. There are classical examples of linear programs that are solved **incrementally**.

- Only a small fraction (sublinear) of the LP needs to be constructed

Did we say something about practicality?

We are solving Linear Programs. There are classical examples of linear programs that are solved **incrementally**.

- Only a small fraction (sublinear) of the LP needs to be constructed
- Example 1: Edmonds' weighted matching algorithm (bah, that is theory only)

Did we say something about practicality?

We are solving Linear Programs. There are classical examples of linear programs that are solved **incrementally**.

- Only a small fraction (sublinear) of the LP needs to be constructed
- Example 1: Edmonds' weighted matching algorithm (bah, that is theory only)
- Example 2: Solving large set-partitioning LPs (airline industry)

Did we say something about practicality?

We are solving Linear Programs. There are classical examples of linear programs that are solved **incrementally**.

- Only a small fraction (sublinear) of the LP needs to be constructed
- Example 1: Edmonds' weighted matching algorithm (bah, that is theory only)
- Example 2: Solving large set-partitioning LPs (airline industry)
- This requires tricks that exploit LP structure. It only works with LPs that have specific, known structure
- That is the case in the above LPs.