Approximate Nonconvex Optimization and Treewidth

Daniel Bienstock\textsuperscript{1}
Triangle Lectures, 2018

\textsuperscript{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

Given:
• $\mathbf{D}$ 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
$$
\min_{f \in F} \frac{1}{D} \sum_{i=1}^{D} \ell(f(\hat{x}_i); \hat{y}_i) + \text{optional regularizer}(f)
$$
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

$$
\min_{f} \frac{1}{D} \sum_{i=1}^{D} \ell(f(\hat{x}^i), \hat{y}^i) + \text{optional regularizer } \Phi(f)
\quad (f \in F, \text{some class})
$$
Function parameterization

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

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

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

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

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

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

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

\(f \in F \) (some class)

Examples:

- **Linear Regression.** \( f(x) = Ax + b \) with \( \ell_2 \)-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.**
  \[
  f(x) = T_{k+1} \circ \sigma \circ T_k \circ \sigma \ldots \circ \sigma \circ T_1(x), \text{ each } T_j \text{ affine.}
  \]
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 $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ to minimize

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

Each $T_i$ affine: $T_i(y) = A_i y + b_i$. Example $(t)$ = $\max f_0; t g$ (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
  \[
  \frac{1}{D} \sum_{i=1}^{D} \|\hat{y}_i - f(\hat{x}_i)\|^2
  \]

- \( f = T_{k+1} \circ \sigma \circ T_k \circ \sigma \ldots \circ \sigma \circ T_1 \)
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 $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ 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 $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ 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
Problem: Compute \( f : \mathbb{R}^n \rightarrow \mathbb{R}^m \) to minimize

\[
\frac{1}{D} \sum_{i=1}^{D} \| \hat{y}_i - f(\hat{x}_i) \|^2
\]
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 \( f = T_{k+1} \circ \sigma \circ T_k \circ \sigma \ldots \circ \sigma \circ T_1 \)
Problem: Compute $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ 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 \textbf{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 $\pm 1$.
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 $f: \mathbb{R}^n \rightarrow \mathbb{R}^m$ 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 $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ 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 $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ 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 $D$
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 $f = T_{k+1} \circ \sigma \circ T_k \circ \sigma \cdots \circ \sigma \circ T_1$
Problem: Compute $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ 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*
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

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

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

$n = \text{dimension of input vectors},$
Exact Training Complexity

**Problem:** Compute $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ 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*

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

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

$n =$ dimension of input vectors, $w =$ size of internal layers
Theorem

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

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

($L$ = largest Lipshitz constant of any $\sigma$, and $N = O(wk)$)
Theorem

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

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

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

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

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

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

provides an $\epsilon$-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 is a parameter that measures how tree-like a graph is.
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** 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 $k$
Informal algorithmic definition:
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. Identify (“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 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 $f(H)$. 

"K-trees", perfect graph literature, (1970s and after)


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 $f(H)$.
- “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 $f(H)$.
- “K-trees”, perfect graph literature, (1970s and after)
Second tool: approximation through digitization

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

$$z \approx \sum_{k=1}^{L} 2^{-k} y_k,$$

where each $y_k$ binary.
Let $0 \leq z \leq 1$ and let $L > 0$ integral
Then we can approximate

$$z \approx \sum_{k=1}^{L} 2^{-k} y_k,$$
where each $y_k$ binary. Error $\leq 2^{-L}$
Let $0 \leq z \leq 1$ and let $L > 0$ integral.

Then we can approximate

$$z \approx \sum_{k=1}^{L} 2^{-k} y_k,$$

where each $y_k$ binary. Error $\leq 2^{-L}$

- Suppose $F : \mathbb{R}^m \rightarrow \mathbb{R}$ with Lipschitz constant $\mathcal{L}$
Let $0 \leq z \leq 1$ and let $L > 0$ integral.
Then we can approximate

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

- Suppose $F : \mathbb{R}^m \rightarrow \mathbb{R}$ with Lipschitz constant $\mathcal{L}$
- Digitize (approximate) each component of $x \in \mathbb{R}^m$
Second tool: approximation through digitization

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

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

- Suppose $F : \mathbb{R}^m \to \mathbb{R}$ with Lipschitz constant $\mathcal{L}$
- Digitize (approximate) each component of $x \in \mathbb{R}^m$
- So $x \to y$ (the binary approximation). $y \in \mathbb{R}^{mL}$
Let $0 \leq z \leq 1$ and let $L > 0$ integral.
Then we can approximate

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

- Suppose $F : \mathbb{R}^m \to \mathbb{R}$ with Lipschitz constant $L$
- Digitize (approximate) each component of $x \in \mathbb{R}^m$
- So $x \to y$ (the binary approximation). $y \in \mathbb{R}^{mL}$
- And $F(x)$ replaced by $\hat{F}(y)$
Let $0 \leq z \leq 1$ and let $L > 0$ integral 
Then we can approximate 

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

- 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 $\hat{F}(y)$ 
- $\hat{F}(y) \leq F(x) + m\mathcal{L} \cdot 2^{-L}$
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, ~ 1990

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

\[ \begin{align*}
\text{Key idea: add new variables that express conjunctions:} \\
\end{align*} \]
Final ingredient: lifted formulations

Lovász and Schrijver, Sherali and Adams, ~ 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, ~ 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 \( x_1x_2(1 - x_3) \)
Final ingredient: lifted formulations

Lovász and Schrijver, Sherali and Adams, ~ 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 \( x_1x_2(1 - x_3) \)
  - 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, ~ 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
    \[
    2x_1 + x_3 + x_4 \leq 1 \text{ is a constraint of } Ax \leq b
    \]
  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, ~ 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 \( x_1x_2(1-x_3) \)
  - 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, ~ 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
    \[ 2x_1 + x_3 + x_4 \leq 1 \]
    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 \(x_1x_2(1 - x_3)\)
  - 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 \(c^Tx \leq d\)
where the subgraph of the intersection graph of \(A\), induced by \(c\)
Final ingredient: lifted formulations

Lovász and Schrijver, Sherali and Adams, ~ 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 \( x_1x_2(1 - x_3) \)
  - 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 \( c^Tx \leq d \)
where the subgraph of the intersection graph of \( A \), induced by \( c \)
has treewidth \( \leq k \)
Approximate optimization of well-behaved functions

Prototype problem:

\[ c^* = \min c^T x \]
\[ \text{s.t. } f_i(x) \leq 0, \quad i = 1, \ldots, m \]
\[ x \in [0, 1]^n \]


**Theorem**

Suppose the intersection graph has tree-width \( \omega \) and let \( \mathcal{L} = \max_i \mathcal{L}_i \).
Approximate optimization of well-behaved functions

Prototype problem:

\[ c^* \triangleq \min c^T x \]

\[ \text{s.t. } f_i(x) \leq 0, \quad i = 1, \ldots, m \]

\[ x \in [0, 1]^n \]


**Theorem**

*Suppose the intersection graph has tree-width \( \omega \) and let \( L = \max_i L_i \). Then, for every \( \epsilon > 0 \) there is an LP relaxation of size*

\[ O \left( \left( \frac{L}{\epsilon} \right)^{\omega+1} n \right) \]

*that guarantees \( \epsilon \)-optimality and \( \epsilon \)-feasibility errors.*
Approximate optimization of well-behaved functions

Prototype problem:

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


**Theorem**

Suppose the intersection graph has tree-width \( \omega \) 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 \( \epsilon \)-optimality and \( \epsilon \)-feasibility errors.

\[ c^T \hat{x} \leq c^* + O(\epsilon), \quad f_i(\hat{x}) \leq O(\epsilon) \quad \forall i \]
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 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 \([-1, 1]^{n+m+N}\).
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 \textit{treewidth} at most \( N + n + m + 1 \)
Thus the LP size given by the expression

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

becomes

\[ O \left( (2\mathcal{L}/\epsilon)^{N+n+m+1} D \right) \]
Thus the LP size given by the expression

\[ O \left( \left( \frac{L}{\epsilon} \right)^{\omega + 1} n \right) \]

becomes

\[ O \left( \left( \frac{2L}{\epsilon} \right)^{N+n+m+1} D \right) \]

The key to linear dependence on \( D \) lies in the fact that the \( D \) does not add to the treewidth.
Thus the LP size given by the expression

\[ O \left( \left( \frac{L}{\epsilon} \right)^{\omega+1} n \right) \]

becomes

\[ O \left( \left( \frac{2L}{\epsilon} \right)^{N+n+m+1} D \right) \]

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

Different architectures → \( N \) and \( \mathcal{L} \).
Architecture-Specific Consequences
Training of Deep Neural Networks with ReLUs

Theorem (Arora, Basu, Mianjy and Mukherjee 2018)

If $k = 1$ (one “hidden layer”) and $m = 1$ there is an exact training algorithm of complexity
Training of Deep Neural Networks with ReLUs

Theorem (Arora, Basu, Mianjy and Mukherjee 2018)

If \( k = 1 \) (one “hidden layer”) and \( m = 1 \) there is an exact training algorithm of complexity

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

Polynomial in the size of the data set, for fixed \( n, w \).
Consequence of our result

If the entries of $A_i; b_i$ are required to be in $[1; 1]$, for any $k; n; m; w; \epsilon$ there is a uniform LP of size $O((nw = \epsilon) k (n + m + N + 1) D)$ with the same guarantees as before.

Core of the proof: In a DNN with $k$ hidden layers the Lipschitz constant of $g(x; y)$ over $[1; 1] n + m + N$ is $nw k$. 


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

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

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 $k, n, m, w, \epsilon$ there is a uniform LP of size

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

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)

Activation units:

\[ y = \begin{cases} 1, & \text{if } a^T z > b_0 \\ \text{otherwise} \end{cases} \]

Network with \( n \) binary inputs, \( m \) binary outputs, \( k \) layers.
The Blum-Rivest setup (Binarized Neural Networks)

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:

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

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

Network with $n$ binary inputs, $m$ binary outputs, $k$ layers
Theorem (Blum and Rivest 1992)

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

\[ \text{Diagram with a neural network structure} \]
Our result on Binarized Neural Networks

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

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

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 \(\epsilon\)
Thank you!
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
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)
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)
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.