Using Integer Programming to Train Neural Networks

Daniel Bienstock, Gonzalo Muñoz, Sebastian Pokutta

Columbia University, Polytechnique Montréal, Georgia Tech

Bordeaux July 2018

Empirical Risk Minimization problem

Empirical Risk Minimization problem

Given:

- $\boldsymbol{D}$ data points $\boldsymbol{(x_i, y_i)}, \quad \boldsymbol{i = 1, \ldots, D}$
- $\boldsymbol{x_i} \in \mathbb{R}^{\boldsymbol{n}}, \; \boldsymbol{y_i} \in \mathbb{R}^{\boldsymbol{m}}$ all i
- A "loss" function $\boldsymbol{\ell} : \mathbb{R}^{\boldsymbol{n}} \times \mathbb{R}^{\boldsymbol{m}} \rightarrow \mathbb{R}$ (not necessarily convex)

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

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

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

Approximate optimization of well-behaved functions

Approximate optimization of well-behaved functions

Prototype problem:

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

Approximate optimization of well-behaved functions

Prototype problem:

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

Each $\boldsymbol{f_i}$ is "well-behaved": Lipschitz constant $\boldsymbol{\mathcal{L}_i}$

Approximate optimization of well-behaved functions

Prototype problem:

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

Each $\boldsymbol{f_i}$ is "well-behaved": Lipschitz constant $\boldsymbol{\mathcal{L}_i}$

Toolset:

- **Intersection graph**

Approximate optimization of well-behaved functions

Prototype problem:

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

Each $\boldsymbol{f_i}$ is "well-behaved": Lipschitz constant $\boldsymbol{\mathcal{L}_i}$

Toolset:

- **Intersection graph**
  A vertex for each variable and an edge whenever two variables appear in the same $\boldsymbol{f_i}$
- **Tree-width**

Approximate optimization of well-behaved functions

Prototype problem:

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

Each $\boldsymbol{f_i}$ is "well-behaved": Lipschitz constant $\boldsymbol{\mathcal{L}_i}$

Toolset:

- **Intersection graph**
  A vertex for each variable and an edge whenever two variables appear in the same $\boldsymbol{f_i}$
- **Tree-width** Min clique number (minus one) over all chordal supergraphs of G

Prototype problem:

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

An extension of work in B. and Muñoz 2015, SIOPT 2018.

Suppose:

the intersection graph has tree-width ω and $\boldsymbol{f_i}$ has Lipschitz constant $\boldsymbol{\mathcal{L}_i} \leq \boldsymbol{\mathcal{L}}$.

Prototype problem:

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

An extension of work in B. and Muñoz 2015, SIOPT 2018.

Suppose:

the intersection graph has tree-width $\boldsymbol{\omega}$ and $\boldsymbol{f_i}$ has Lipschitz constant $\boldsymbol{\mathcal{L}_i} \le \boldsymbol{\mathcal{L}}$.

Then, for every $\mathbf{0} < \boldsymbol{\epsilon} < \mathbf{1}$ there is an **LP** relaxation with

Prototype problem:

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

An extension of work in B. and Muñoz 2015, SIOPT 2018.

Suppose:

the intersection graph has tree-width ω and f_i has Lipschitz constant $\mathcal{L}_i \leq \mathcal{L}$.

Then, for every $0 < \epsilon < 1$ there is an **LP** relaxation with

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

Prototype problem:

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

An extension of work in B. and Muñoz 2015, SIOPT 2018.

Suppose:

the intersection graph has tree-width ω and $\boldsymbol{f_i}$ has Lipschitz constant $\boldsymbol{\mathcal{L}_i} \leq \boldsymbol{\mathcal{L}}$.

Then, for every $\mathbf{0} < \boldsymbol{\epsilon} < \mathbf{1}$ there is an **LP** relaxation with

$\boldsymbol{O\left((\mathcal{L}/\epsilon)^{\omega+1}\, n \log(\mathcal{L}/\epsilon)\right)}$ variables and constraints

Optimality and feasibility errors $\boldsymbol{O(\epsilon)}$

Application to ERM problem

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

Application to ERM problem

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

Linearize objective using epigraph trick

$$\min_{f \in F} \frac{1}{D} \sum_{i=1}^{D} L_i$$

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

Function parameterization

$$\min_{f \in F} \frac{1}{D} \sum_{i=1}^{D} L_i$$

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

Examples:

- **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 $\boldsymbol{T_j}$ affine, **parameterized** by its coefficients.
- **Linear Regression.** $f(x) = Ax + b$ with ℓ_2-loss.
- **Binary Classification.** Varying f architectures and cross-entropy loss: $\ell(p, y) = -y \log(p) - (1 - y) \log(1 - p)$

Function parameterization

$$\min_{f \in F} \frac{1}{D} \sum_{i=1}^{D} L_i$$

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

Examples:

- **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 $\boldsymbol{T_j}$ affine, **parameterized** by its coefficients.
- **Linear Regression.** $f(x) = Ax + b$ with ℓ_2-loss.
- **Binary Classification.** Varying f architectures and cross-entropy loss: $\ell(p, y) = -y \log(p) - (1 - y) \log(1 - p)$

We assume $F = \{f(x, \theta) \text{ for } \theta \in \Theta \subseteq [-1, 1]^N\}$

We now apply the LP approximation result to:

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

Let θ^* be an optimal solution. For every $\mathbf{0} < \epsilon < \mathbf{1}$ there is an **LP** of size

$$\boldsymbol{O\left((\mathcal{L}/\epsilon)^{2N+2}\,(N+D)\log(\mathcal{L}/\epsilon)\right)}$$

We now apply the LP approximation result to:

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

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

Let θ^* be an optimal solution. For every $\mathbf{0} < \epsilon < \mathbf{1}$ there is an **LP** of size

$$\boldsymbol{O\left((\mathcal{L}/\epsilon)^{2N+2}\,(N+D)\log(\mathcal{L}/\epsilon)\right)}$$

such that its optimal solution $\hat{\theta}$ satisfies:

$$\frac{1}{D} \sum_{i=1}^{D} \ell(f(x^i, \hat{\theta}), y^i) \leq \frac{1}{D} \sum_{i=1}^{D} \ell(f(x^i, \theta^*), y^i) + \epsilon$$

We now apply the LP approximation result to:

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

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

Let θ^* be an optimal solution. For every $\mathbf{0} < \epsilon < \mathbf{1}$ there is an **LP** of size

$$\boldsymbol{O\left((\mathcal{L}/\epsilon)^{2N+2}\,(N+D)\log(\mathcal{L}/\epsilon)\right)}$$

such that its optimal solution $\hat{\theta}$ satisfies:

$$\frac{1}{D} \sum_{i=1}^{D} \ell(f(x^i, \hat{\theta}), y^i) \leq \frac{1}{D} \sum_{i=1}^{D} \ell(f(x^i, \theta^*), y^i) + \epsilon$$

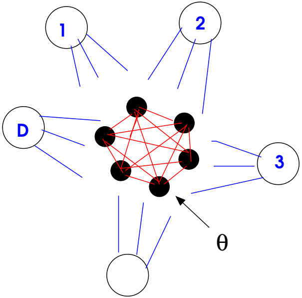
$\mathcal{L}$ is an upper bound on the Lipschitz constant of $g^i(\theta) \doteq \ell(f(x^i, \theta), y^i)$.

Proof sketch:

$$\min_{\theta \in \Theta} \frac{1}{D} \sum_{i=1}^{D} L_i, \qquad \text{s.t.} \quad L_i \geq \ell(f(x^i, \theta), y^i) \quad 1 \leq i \leq D$$

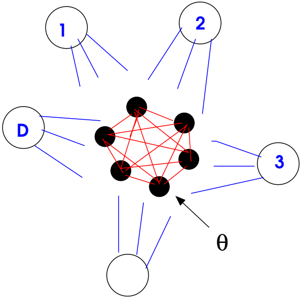
Proof sketch:

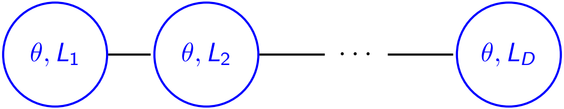
$$\min_{\theta \in \Theta} \frac{1}{D} \sum_{i=1}^{D} L_i, \qquad \text{s.t.} \quad L_i \geq \ell(f(x^i, \theta), y^i) \quad 1 \leq i \leq D$$



Proof sketch:

$$\min_{\theta \in \Theta} \frac{1}{D} \sum_{i=1}^{D} L_i, \qquad \text{s.t.} \quad L_i \geq \ell(f(x^i, \theta), y^i) \quad 1 \leq i \leq D$$





Application: training of Deep Neural Networks with ReLUs

Application: training of Deep Neural Networks with ReLUs

As per Arora Basu Mianjy Mukherjee ICLR '18

The setup:

- D data points (x_i, y_i), $1 \leq i \leq D$, $x_i \in \mathbb{R}^n$, $y_i \in \mathbb{R}$

Application: training of Deep Neural Networks with ReLUs

As per Arora Basu Mianjy Mukherjee ICLR '18

The setup:

- $\boldsymbol{D}$ data points $(\boldsymbol{x_i}, \boldsymbol{y_i})$, $\mathbf{1} \leq \boldsymbol{i} \leq \boldsymbol{D}$, $\boldsymbol{x_i} \in \mathbb{R}^{\boldsymbol{n}}$, $\boldsymbol{y_i} \in \mathbb{R}$
- **Task:** **compute** a function $\boldsymbol{f} : \mathbb{R}^{\boldsymbol{n}} \rightarrow \mathbb{R}$ to minimize

$$\frac{1}{D} \sum_{i=1} (y_i - f(x_i))^2$$

Application: training of Deep Neural Networks with ReLUs

As per Arora Basu Mianjy Mukherjee ICLR '18

The setup:

- $\boldsymbol{D}$ data points $\boldsymbol{(x_i, y_i)}$, $\boldsymbol{1 \leq i \leq D}$, $\boldsymbol{x_i} \in \mathbb{R}^{\boldsymbol{n}}$, $\boldsymbol{y_i} \in \mathbb{R}$
- **Task:** **compute** a function $\boldsymbol{f} : \mathbb{R}^{\boldsymbol{n}} \rightarrow \mathbb{R}$ to minimize

$$\frac{1}{D} \sum_{i=1} (y_i - f(x_i))^2$$

- $\boldsymbol{f = T_{k+1} \circ \sigma \circ T_k \circ \sigma \ldots \circ \sigma \circ T_1}$ ("$\circ$" $=$ composition)

Application: training of Deep Neural Networks with ReLUs

As per Arora Basu Mianjy Mukherjee ICLR '18

The setup:

- $\boldsymbol{D}$ data points $(\boldsymbol{x_i}, \boldsymbol{y_i})$, $\boldsymbol{1} \le \boldsymbol{i} \le \boldsymbol{D}$, $\boldsymbol{x_i} \in \mathbb{R}^{\boldsymbol{n}}$, $\boldsymbol{y_i} \in \mathbb{R}$
- **Task:** **compute** a function $\boldsymbol{f} : \mathbb{R}^{\boldsymbol{n}} \to \mathbb{R}$ to minimize

$$\frac{1}{D} \sum_{i=1} (y_i - f(x_i))^2$$

- $\boldsymbol{f} = \boldsymbol{T_{k+1}} \circ \boldsymbol{\sigma} \circ \boldsymbol{T_k} \circ \boldsymbol{\sigma} \ldots \circ \boldsymbol{\sigma} \circ \boldsymbol{T_1}$ ("$\circ$" = composition)
- $\boldsymbol{\sigma(t) = \max\{0, t\}}$

Application: training of Deep Neural Networks with ReLUs

As per Arora Basu Mianjy Mukherjee ICLR '18

The setup:

- $\boldsymbol{D}$ data points $(\boldsymbol{x_i}, \boldsymbol{y_i})$, $\boldsymbol{1} \leq \boldsymbol{i} \leq \boldsymbol{D}$, $\boldsymbol{x_i} \in \mathbb{R}^{\boldsymbol{n}}$, $\boldsymbol{y_i} \in \mathbb{R}$
- **Task: compute** a function $\boldsymbol{f} : \mathbb{R}^{\boldsymbol{n}} \rightarrow \mathbb{R}$ to minimize

$$\frac{1}{D} \sum_{i=1} (y_i - f(x_i))^2$$

- $\boldsymbol{f} = \boldsymbol{T_{k+1}} \circ \boldsymbol{\sigma} \circ \boldsymbol{T_k} \circ \boldsymbol{\sigma} \ldots \circ \boldsymbol{\sigma} \circ \boldsymbol{T_1}$ ("$\circ$" = composition)
- $\boldsymbol{\sigma(t)} = \mathbf{max}\{\mathbf{0}, \boldsymbol{t}\}$
- Each $\boldsymbol{T_h}$ affine: $\boldsymbol{T_h(y) = A_h y + b_h}$,

Application: training of Deep Neural Networks with ReLUs

As per Arora Basu Mianjy Mukherjee ICLR '18

The setup:

- $\boldsymbol{D}$ data points $(\boldsymbol{x_i}, \boldsymbol{y_i})$, $\mathbf{1} \le \boldsymbol{i} \le \boldsymbol{D}$, $\boldsymbol{x_i} \in \mathbb{R}^{\boldsymbol{n}}$, $\boldsymbol{y_i} \in \mathbb{R}$
- **Task:** **compute** a function $\boldsymbol{f} : \mathbb{R}^{\boldsymbol{n}} \to \mathbb{R}$ to minimize

$$\frac{1}{D} \sum_{i=1} (y_i - f(x_i))^2$$

- $\boldsymbol{f} = \boldsymbol{T_{k+1}} \circ \boldsymbol{\sigma} \circ \boldsymbol{T_k} \circ \boldsymbol{\sigma} \ldots \circ \boldsymbol{\sigma} \circ \boldsymbol{T_1}$ ("$\circ$" = composition)
- $\boldsymbol{\sigma(t)} = \max\{0, \boldsymbol{t}\}$
- Each $\boldsymbol{T_h}$ affine: $\boldsymbol{T_h(y) = A_h y + b_h}$,
- For some $\boldsymbol{w}$, $\boldsymbol{A_1}$ is $\boldsymbol{n} \times \boldsymbol{w}$, $\boldsymbol{A_{k+1}}$ is $\boldsymbol{w} \times \mathbf{1}$, $\boldsymbol{A_h}$ is $\boldsymbol{w} \times \boldsymbol{w}$ otherwise.

Application: training of Deep Neural Networks with ReLUs

As per Arora Basu Mianjy Mukherjee ICLR '18

The setup:

- $\boldsymbol{D}$ data points $(\boldsymbol{x_i}, \boldsymbol{y_i})$, $\boldsymbol{1 \le i \le D}$, $\boldsymbol{x_i} \in \mathbb{R}^{\boldsymbol{n}}$, $\boldsymbol{y_i} \in \mathbb{R}$
- **Task:** **compute** a function $\boldsymbol{f} : \mathbb{R}^{\boldsymbol{n}} \to \mathbb{R}$ to minimize

$$\frac{1}{D} \sum_{i=1} (y_i \, - \, f(x_i))^2$$

- $\boldsymbol{f \;=\; T_{k+1} \circ \sigma \circ T_k \circ \sigma \ldots \circ \sigma \circ T_1}$ ("$\circ$" $=$ composition)
- $\boldsymbol{\sigma(t) \;=\; \max\{0, t\}}$
- Each $\boldsymbol{T_h}$ affine: $\boldsymbol{T_h(y) = A_h y + b_h}$,
- For some $\boldsymbol{w}$, $\boldsymbol{A_1}$ is $\boldsymbol{n \times w}$, $\boldsymbol{A_{k+1}}$ is $\boldsymbol{w \times 1}$, $\boldsymbol{A_h}$ is $\boldsymbol{w \times w}$ otherwise. Similarly with the $\boldsymbol{b_h}$.

- $\boldsymbol{D}$ data points $(\boldsymbol{x_i}, \boldsymbol{y_i})$, $\boldsymbol{1 \leq i \leq D}$, $\boldsymbol{x_i} \in \mathbb{R}^{\boldsymbol{n}}$, $\boldsymbol{y_i} \in \mathbb{R}$
- **Task:** **compute** a function $\boldsymbol{f} : \mathbb{R}^{\boldsymbol{n}} \rightarrow \mathbb{R}$ to minimize

$$\frac{1}{D} \sum_{i=1}^{D} (y_i - f(x_i))^2$$

- $\boldsymbol{f = T_{k+1} \circ \sigma \circ T_k \circ \sigma \ldots \circ \sigma \circ T_1}$ ("$\circ$" = composition)
- $\boldsymbol{\sigma(t) = \max\{0, t\}}$
- Each $\boldsymbol{T_h}$ affine: $\boldsymbol{T_h(y) = A_h y + b_h}$,
- For some $\boldsymbol{w}$, $\boldsymbol{A_1}$ is $\boldsymbol{n} \times \boldsymbol{w}$, $\boldsymbol{A_{k+1}}$ is $\boldsymbol{w} \times \boldsymbol{1}$, $\boldsymbol{A_h}$ is $\boldsymbol{w} \times \boldsymbol{w}$ otherwise. Similarly with the $\boldsymbol{b_h}$.

- $\boldsymbol{D}$ data points $\boldsymbol{(x_i, y_i)}$, $\boldsymbol{1 \leq i \leq D}$, $\boldsymbol{x_i \in \mathbb{R}^n}$, $\boldsymbol{y_i \in \mathbb{R}}$
- **Task:** **compute** a function $\boldsymbol{f} : \mathbb{R}^{\boldsymbol{n}} \rightarrow \mathbb{R}$ to minimize

$$\frac{1}{D} \sum_{i=1}^{D} (y_i - f(x_i))^2$$

- $\boldsymbol{f = T_{k+1} \circ \sigma \circ T_k \circ \sigma \ldots \circ \sigma \circ T_1}$ (“$\circ$” = composition)
- $\boldsymbol{\sigma(t) = \max\{0, t\}}$
- Each $\boldsymbol{T_h}$ affine: $\boldsymbol{T_h(y) = A_h y + b_h}$,
- For some $\boldsymbol{w}$, $\boldsymbol{A_1}$ is $\boldsymbol{n \times w}$, $\boldsymbol{A_{k+1}}$ is $\boldsymbol{w \times 1}$, $\boldsymbol{A_h}$ is $\boldsymbol{w \times w}$ otherwise. Similarly with the $\boldsymbol{b_h}$.

Theorem (Arora et al 2018).
If k = 1 (one “hidden layer”) there is an exact algorithm of complexity

- $\boldsymbol{D}$ data points $(\boldsymbol{x_i}, \boldsymbol{y_i})$, $\mathbf{1} \leq \boldsymbol{i} \leq \boldsymbol{D}$, $\boldsymbol{x_i} \in \mathbb{R}^{\boldsymbol{n}}$, $\boldsymbol{y_i} \in \mathbb{R}$
- **Task:** **compute** a function $\boldsymbol{f} : \mathbb{R}^{\boldsymbol{n}} \rightarrow \mathbb{R}$ to minimize

$$\frac{1}{D} \sum_{i=1}^{D} (y_i - f(x_i))^2$$

- $\boldsymbol{f} = \boldsymbol{T_{k+1}} \circ \boldsymbol{\sigma} \circ \boldsymbol{T_k} \circ \boldsymbol{\sigma} \ldots \circ \boldsymbol{\sigma} \circ \boldsymbol{T_1}$ ("$\circ$" = composition)
- $\boldsymbol{\sigma(t)} = \mathbf{max}\{\mathbf{0}, \boldsymbol{t}\}$
- Each $\boldsymbol{T_h}$ affine: $\boldsymbol{T_h(y)} = \boldsymbol{A_h y} + \boldsymbol{b_h}$,
- For some $\boldsymbol{w}$, $\boldsymbol{A_1}$ is $\boldsymbol{n} \times \boldsymbol{w}$, $\boldsymbol{A_{k+1}}$ is $\boldsymbol{w} \times \mathbf{1}$, $\boldsymbol{A_h}$ is $\boldsymbol{w} \times \boldsymbol{w}$ otherwise. Similarly with the $\boldsymbol{b_h}$.

Theorem (Arora et al 2018).
If k = 1 (one "hidden layer") there is an exact 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 $\boldsymbol{n}, \boldsymbol{w}$

Our result: if the entries of A_i, b_i are required to be in $[-1, 1]$, for any $\boldsymbol{k}, \boldsymbol{n}, \boldsymbol{w}, \epsilon$ there is an LP of size

$$\boldsymbol{O\left((w/\epsilon)^{poly(n,k,w)}\left(poly(n,k,w)+D\right)\log(w/\epsilon)\right)}$$

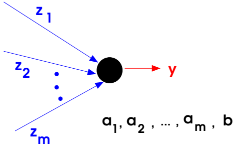
Our result: if the entries of A_i, b_i are required to be in $[-1,1]$, for any $\boldsymbol{k}, \boldsymbol{n}, \boldsymbol{w}, \epsilon$ there is an LP of size

$$\boldsymbol{O\left((w/\epsilon)^{poly(n,k,w)}\,(poly(n,k,w)+D)\log(w/\epsilon)\right)}$$

- $poly(n,k,w) =$ quadratic in w, in k, linear in n
- Treewidth **independent** of D
- Number of variables **linear** in D

The Arora-Blum setup (Binarized Neural Networks)

Activation units:

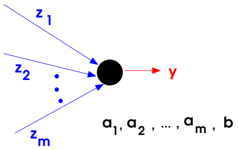


With $\boldsymbol{z} \in \{\mathbf{0}, \mathbf{1}\}^{\boldsymbol{m}}$,

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

The Arora-Blum setup (Binarized Neural Networks)

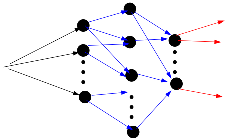
Activation units:



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



Binarized Neural Networks, 2

Training data: Set of D pairs (x^i, y^i), $1 \leq i \leq D$

$x^i \in \{0,1\}^n, \quad y^i \in \{0,1\}^m$

Binarized Neural Networks, 2

Training data: Set of D pairs (x^i, y^i), $1 \leq i \leq D$
$x^i \in \{0,1\}^n, \quad y^i \in \{0,1\}^m$

Problem: Compute the activation function at each node to

$$\min \ \frac{1}{D} \sum_{i=1}^{D} \ell(f(x^i), y^i)$$

(f = network function)

- When $\ell \in$ (absolute value, 2-norm squared) NP-hard if $k = 3$ and $D = n$, $m = 1$
- But we are interested in the case D very large compared to n
- And also other loss functions, e.g. smooth convex

Our result on Binarized Neural Networks

Training data: Set of D pairs (x^i, y^i), $1 \leq i \leq D$

$$x^i \in \{0,1\}^n, \quad y^i \in \{0,1\}^m$$

Our result on Binarized Neural Networks

Training data: Set of D pairs (x^i, y^i), $1 \leq i \leq D$
$x^i \in \{0,1\}^n, \quad y^i \in \{0,1\}^m$

ERM problem: Compute the activation function at each node to

$$\min \ \frac{1}{D} \sum_{i=1}^{D} \ell(f(x^i), y^i)$$

(f = network function)

Theorem. When $\ell \in$ (absolute value, 2-norm squared) there is an LP of encoding length

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

that solves the ERM problem exactly in absolute case, and within $O(\epsilon)$ additive error in the 2-norm case.

Extensions