Efficient and robust optimization methods for training binarized deep neural networks

Bubacarr Bah

German Research Chair of Mathematics in Data Science, AIMS S. Africa

Associate Professor and Head of Data Science, MRC Unit The Gambia

65th Annual SAMS Congress

Stellenbosh University, South Africa

December 06 – 08, 2022

AIMS African Institute for Mathematical Sciences SOUTH AFRICA

UNIVERSITEIT STELLENBOSCH UNIVERSITY

MRC Unit The Gambia

LONDON SCHOOL of HYGIENE &TROPICAL MEDICINE

Sponsors:

Bundesministerium für Bildung und Forschung

Unterstützt von / Supported by Alexander von Humboldt Stiftung/Foundation

DAAD

Outline of talk

Introduction

Deep Neural Networks (DNN)

Binary Deep Neural Networks (BDNN)

Training BDNN

Variants of BDNN Model

Computations

Conclusion

- Joint work with **Jannis Kurtz** at Siegen University, Germany

An Integer Programming Approach to Deep Neural Networks with Binary Activation Functions

Jannis Kurtz [1] **Bubacarr Bah** [2 3]

Abstract

We study deep neural networks with binary activation functions (BDNN), i.e. the activation function only has two states. We show that the BDNN can be reformulated as a mixed-integer linear program which can be solved to global optimality by classical integer programming solvers. Additionally, a heuristic solution algorithm is presented and we study the model under data uncertainty, applying a two-stage robust optimization approach. We implemented our methods on random and real datasets and show that the heuristic version of the BDNN outperforms classical deep neural networks on the Breast Cancer Wisconsin dataset while performing worse on random data.

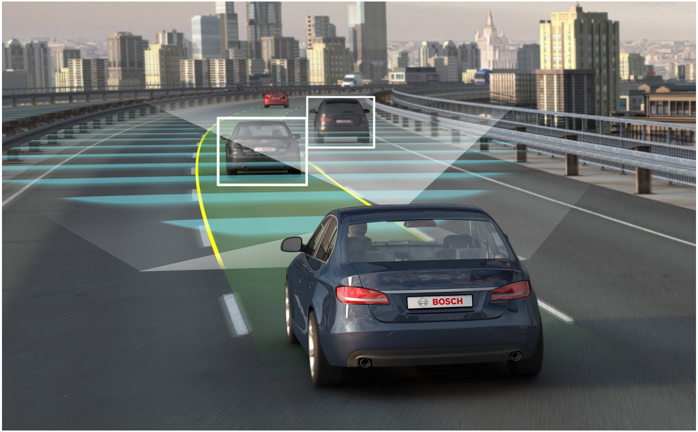
to be more robust to adversarial perturbations than the continuous activation networks (Qin et al., 2020). Furthermore low-powered computations may benefit from discrete activations as a form of coarse quantizations (Plagianakos et al., 2001; Bengio et al., 2013; Courbariaux et al., 2015; Rastegari et al., 2016). Nevertheless, gradient descent-based training behaves like a black box, raising a lot of questions regarding the explainability and interpretability of internal representations (Hampson & Volper, 1990; Plagianakos et al., 2001; Bengio et al., 2013).

On the other hand, integer programming (IP) is known as a powerful tool to model a huge class of real-world optimization problems (Wolsey, 1998). Recently it was successfully applied to machine learning problems involving sparsity constraints and to evaluate trained neural networks (Bertsimas et al., 2017; 2019b; Fischetti & Jo, 2018).

Introduction

Deep Learning – Success Stories

- Neural networks/**deep learning** revolutionized Machine Learning



Self-Driving Cars

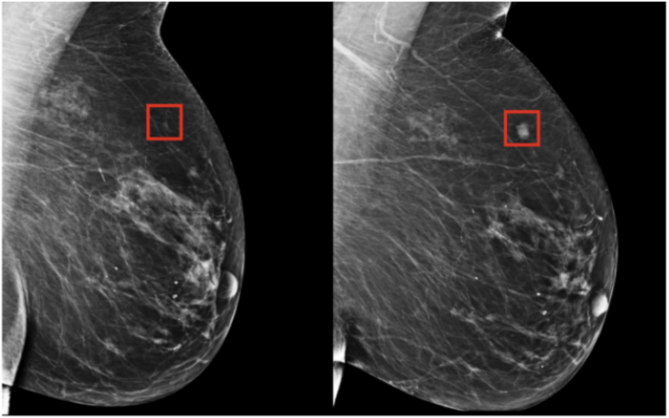
Deep Learning – Success Stories

- Neural networks/**deep learning** revolutionized Machine Learning

Machine Translation

Deep Learning – Success Stories

- Neural networks/**deep learning** revolutionized Machine Learning



Medical Diagnostics

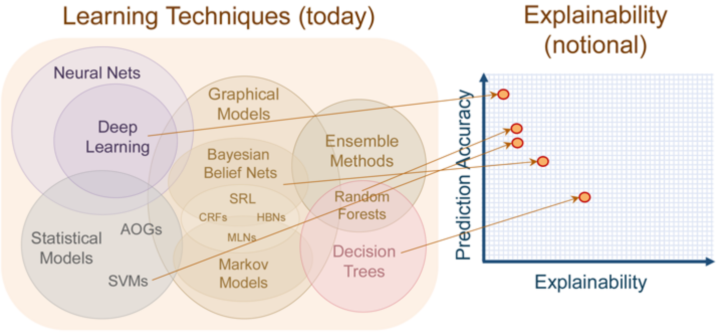
Deep Learning – Challenges

DL presents many opportunities but it is confronted with many **challenges**

- **Absence of Mathematics**: Ingrid Daubechies (Duke University) writes in a recent article that "*Machine learning works spectacularly well, but mathematicians aren't quite sure why*"
- **High-dimensionality**: "High-dimensional Data Analysis: The Curses and Blessings of Dimensionality"
- **Heterogeneity and Incompleteness**: structured and unstructured data: numeric; pictures; videos; text, etc. Missing data points.
- **Scale**: massive datasets instead of small samples that statisticians normally deal with.
- **Timeliness**: an elegant theorem which takes a longer time to prove might be less useful than a medium-quality ("quick-and-dirty") solution to a pressing problem that requires instantaneous decision-making.
- **Ethics**: privacy, security, bias & many other ethical issues of concern.
- ...

Deep Learning – Challenges

- **Explainability**: DL algorithms (ML algorithms in general) are **black boxes**

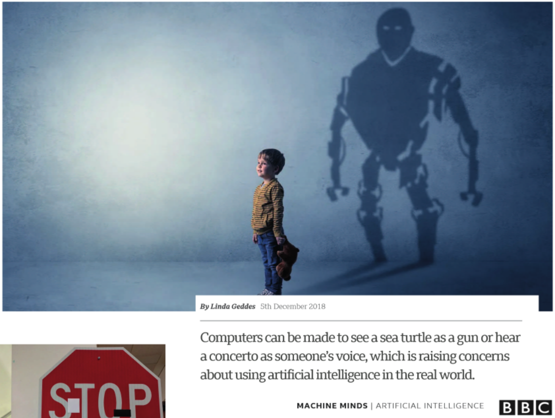


http : nautil.usissue40learningis−artificial−intelligence−permanently−inscrutable

- **This work** attempts to find a solution to this problem

Deep Learning – Challenges

- **Robustness**: DL algorithms are **not robust** to perturbations/noise



By Linda Geddes 5th December 2018

Computers can be made to see a sea turtle as a gun or hear a concerto as someone's voice, which is raising concerns about using artificial intelligence in the real world.

MACHINE MINDS | ARTIFICIAL INTELLIGENCE BBC

Courtesy: Gitta Kutyniok

- **This work** attempts to find a solution to this problem too

Supervised Learning

- Data $\mathcal{D} = \left\{x^i, y^i\right\}_{i=1}^m$: features $x^i \in \mathbb{R}^N$, targets/labels $y^i \in \mathbb{R}$
- Model (function/map/hypothesis) $h_\theta(\mathbf{x})$ satisfying

$$y^i = h_\theta(x^i) + \epsilon^i, \quad \Rightarrow \text{ predictions } \hat{y}^i = h_\theta(x^i)$$

parameter vector $\theta = (\theta_0, \theta_1, \ldots, \theta_n)$, noise ϵ^i

- Linear model (example)

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \cdots + \theta_n x_n = \theta^T x = \langle \theta, x \rangle, \quad (x_0 = 1)$$

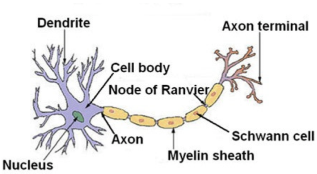
- Metric/distance (loss function $\ell(\cdot)$), e.g. $\ell(\mathbf{z}) = \|z\|_p^p = \sum_{j=1}^n |z_j|^p$

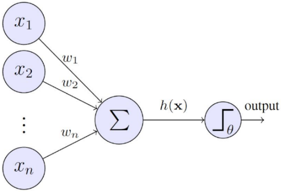
$$\mathcal{L}(\theta) = \frac{1}{m} \sum_{i=1}^m \ell\left(h_\theta\left(x^i\right), y^i\right)$$

Deep Neural Networks (DNN)

Deep Neural Networks

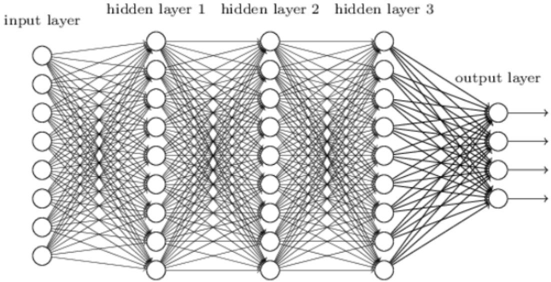
- Neural networks (NN) are (**non-linear**) ML models/algorithms
- Neural networks inspired by how **brain** process information





McCulloch and Pitts Neuron [1943]

Deep Neural Networks



- Deep learning is done by solving for the optimal parameters in

$$\min_{\theta} \ \mathcal{L}(\theta) = \frac{1}{m} \sum_{i=1}^{m} \ell\left(y^{i}, \hat{y}^{i}\right)$$

- DNN are trained through forward propagation and back propagation
- Typically, the **crucial** back propagation is done via Gradient Descent

$$\theta_{t+1} = \theta_t - \eta \nabla_{\theta} \mathcal{L}\left(\theta_t\right), \quad t = 0, 1, \ldots$$

Deep Neural Networks

Non-linear model

$$\hat{y} = h_\theta(\mathbf{x}) = \sigma^K\left(W^K\sigma^{K-1}\left(W^{K-1}\cdots W^2\sigma^1\left(W^1x\right)\right)\right)$$

- Data point $x \in \mathbb{R}^N$
- Number of layers K
- Weight matrices $W^k \in \mathbb{R}^{d_k \times d_{k-1}}$
- Width of the k-th layer is d_k
- Parameter set $\theta = \left\{W^i\right\}_{i=1}^K$
- Activation function $\sigma^k\ :\ \mathbb{R} \to \mathbb{R}$ (applied componentwise)

Supervised Deep Learning

Mathematical framework

$$\min \sum_{i=1}^{m} \ell\left(y^i, \hat{y}^i\right)$$

$$\text{s.t. } \hat{y}^i = \sigma^K\left(W^K \sigma^{K-1}\left(W^{K-1} \cdots W^2 \sigma^1\left(W^1 x^i\right)\right)\right) \quad i \in [m]$$

$$W^k \in \mathbb{R}^{d_k \times d_{k-1}} \quad \forall k \in [K]$$

In the classification setting for example

- Labelled training data $\left(x^1, y^1\right), \ldots, (x^m, y^m) \in \mathbb{R}^N \times \{0,1\}$
- Loss function $\ell \ : \ \{0,1\} \times \mathbb{R}^{d_K} \to \mathbb{R}$

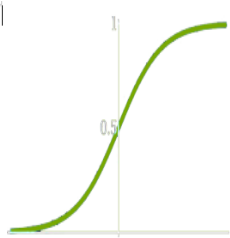
loss function examples

1. ℓ_p-norms $\|y^i - \hat{y}^i\|_p$
2. Cross entropy $-\left(y^i \log\left(\hat{y}^i\right) + \left(1 - y^i\right) \log\left(1 - \hat{y}^i\right)\right)$

Activation Function

- There a many types of activation function including the following

(a) sigmoid

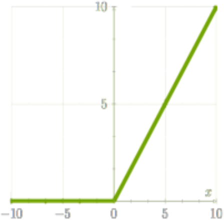


$\sigma(\alpha) = \frac{1}{1+e^{(-\alpha)}}$

(b) tanh

$\sigma(\alpha) = \frac{e^{(\alpha)}-e^{(-\alpha)}}{e^{(\alpha)}+e^{(-\alpha)}}$

(c) ReLU



$\sigma(\alpha) = \max\{0, \alpha\}$

Binary Deep Neural Networks(BDNN)

Binary Deep Neural Networks(BDNN)

BDNN

- Activation function: $\sigma^k(\alpha) = \begin{cases} 0 & \text{if } \alpha < 0 \\ 1 & \text{otherwise} \end{cases}$
- $W^k \in \{0,1\}^{d_k \times d_{k-1}}$ $\left(\text{or } W^k \in \mathbb{R}^{d_k \times d_{k-1}}\right)$

Properties of BDNN

- consume less memory
- be more robust against noise/adversarial attacks
- be less accurate / less complex

Training BDNN

Training BDNN

Gradient-based Methods

- **Forward propagation**: stochastic binarization of weights
- **Back propagation**: approximate the binary activation function by continuous function, e.g.

$$\sigma(\alpha) = \max\{-1, \min\{1, \alpha\}\}$$

Mixed-integer Programming Methods

- Evaluation of trained DNN with ReLU activation using MILP
  [Fischetti, Jo (2018)]
- Calculate adversarial samples for trained DNN / BDNN using MILP
  [Khalil et al. (2019)]
- Train BDNN using MILP
  [Icarte et al. (2019)]

Mixed-Integer Programming Formulation

Theorem (B. & Kurtz (2020))

Training a BDNN, i.e. solving the following problem

$$\begin{aligned} \min \quad & \sum_{i=1}^{m} \mathbb{I}_{\{y^i \neq \hat{y}^i\}} \\ \text{s.t.} \quad & \hat{y}^i = \sigma^K \left(W^K \sigma^{K-1} \left(W^{K-1} \cdots W^2 \sigma^1 \left(W^1 x^i \right) \right) \right) \quad i \in [m] \\ & W^k \in \mathbb{R}^{d_k \times d_{k-1}} \quad \forall k \in [K] \end{aligned} \tag{1}$$

can be done by solving a mixed-integer linear programming (MILP) formulation of polynomial size.

Remark

The weights can be chosen real or binary.

Proof I

Problem (1) is equivalent to

$$
\begin{aligned}
\min \quad & \sum_{i:y^i=0} u^{i,K} + \sum_{i:y^i=1} \left(1 - u^{i,K}\right) \\
\text{s.t.} \quad & W^1 x^i < M_1 u^{i,1} \\
& W^1 x^i \geq M_1 \left(u^{i,1} - 1\right) \\
& W^k x^i < M_k u^{i,k} \quad \forall k \in [K] \setminus \{1\} \\
& W^k x^i \geq M_k \left(u^{i,k} - 1\right) \quad \forall k \in [K] \setminus \{1\} \\
& W^k \in [-1,1]^{d_k \times d_{k-1}} \quad \forall k \in [K] \\
& u^{i,k} \in \{0,1\}^{d_k} \quad \forall k \in [K],\ i \in [m]
\end{aligned}
$$

where $M_1 := d_0 r + 1$ and $M_k := d_{k-1} + 1$

Proof II

The **quadratic terms** $w^k_{\ell j} u^{i,k-1}_j$ can be replaced by a new variable $s^{i,k}_{\ell j} \in [-1, 1]$ and the **equality**

$$w^k_{\ell j} u^{i,k-1}_j = s^{i,k}_{\ell j}$$

is ensured by adding the constraints

$$\begin{aligned}
s^{i,k}_{\ell j} &\leq u^{i,k}_j \\
s^{i,k}_{\ell j} &\geq -u^{i,k}_j \\
s^{i,k}_{\ell j} &\leq w^k_{\ell j} + \left(1 - u^{i,k}_j\right) \\
s^{i,k}_{\ell j} &\geq w^k_{\ell j} - \left(1 - u^{i,k}_j\right)
\end{aligned}$$

This concludes the proof.

MILP Solution Methods

Algorithms

The MILP formulation can be solved as follows:

- to global optimality by classical IP solvers as CPLEX or Gurobi
- using exact methods: branch & bound method or cutting plane/decomposition methods
- using heuristics like the mountain-climbing procedure: iteratively optimize over the u- and the W-variables in the quadratic formulation

Algorithmic Details

- Data points can iteratively be added to the formulation
- Integer programming methods often provide optimality gaps
- Number of integer variables bounded by $\mathcal{O}(dKm)$ (linear in the number of data points)
- Integer programming formulations are hard to solve!
- Especially with Big-M constraints!

Variants of BDNN Model

Model Variants

Variants of the MILP Model

- regression variants
- quadratic loss functions
- add regularizers
- multiclass classification
- more general binary activation functions

$$\sigma^k(\alpha) = \begin{cases} \beta_k & \text{if } \alpha < \lambda_k \\ \gamma_k & \text{otherwise} \end{cases}$$

  where λ_k can be trained.
- more general discrete activation functions can be used:

$$\sigma^k(\alpha) = v \quad \text{if } \underline{\lambda_k^v} \leq \alpha \leq \overline{\lambda_k^v}, \quad v \in V \subset \mathbb{Z}$$

- sparsity constraints can be added
- robust optimization approaches to model uncertainty in the data

Robust Optimization

Enforce robustness during training

- Given an uncertainty set $U := U^1 \times \cdots \times U^m$ where

$$U^i = \left\{ \delta \in \mathbb{R}^N \mid \|\delta\| \leq r_i \right\},$$

- the two-stage robust counterpart of the BDNN formulation is

$$\min_{W^k} \max_{\delta \in U} \min_{u^{i,k}} \left\{ \sum_{i=1}^{m} \ell\left(y^i, u^{i,K}\right) \ : \ W^1, \ldots, W^K, u^{1,1}, \ldots, u^{m,K} \in P(X, \delta) \right\}$$

where $P(X, \delta)$ is the set of feasible solutions of the inequality system

$$\begin{aligned}
& W^1 \left(x^i + \delta^i\right) < M_1 u^{i,1} \\
& W^1 \left(x^i + \delta^i\right) \geq M_1 \left(u^{i,1} - 1\right) \\
& W^k x^i < M_k u^{i,k} \quad \forall k \in [K] \setminus \{1\} \\
& W^k x^i \geq M_k \left(u^{i,k} - 1\right) \quad \forall k \in [K] \setminus \{1\} \\
& W^k \in [-1, 1]^{d_k \times d_{k-1}} \quad \forall k \in [K] \\
& u^{i,k} \in \{0, 1\}^{d_k} \quad \forall k \in [K],\ i \in [m]
\end{aligned}$$

Computations

Details of Neural Networks

DNN

```
model = Sequential()
model.add(Dense(units=shape[1], activation='relu', input_dim=N, use_bias=False))
model.add(Dense(units=n_classes, activation='softmax', use_bias=False))

opt = Adam(lr=LR)
model.compile(loss="binary_crossentropy", optimizer=opt, metrics=metric)
model.summary()
```

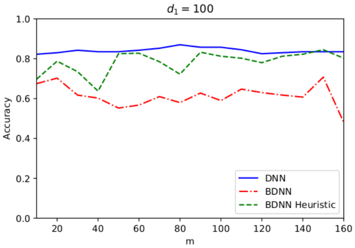
```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 100)               10000
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 200
=================================================================
Total params: 10,200
Trainable params: 10,200
Non-trainable params: 0
_________________________________________________________________
```

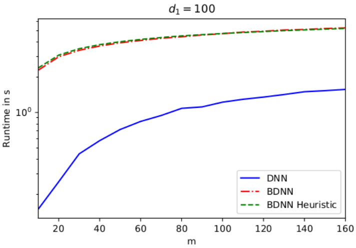
BDNN – same architecture as DNN

1. Direct solvers (Gurobi) - **BDNN**
2. Heuristic (mountain-climbing) – **BDNN Heuristic**

Random Data

- **Data points**: vectors of dimension $N = 100$ with entries drawn uniformly at random from 2 overlapping regions of $\mathbb{R}$
- **Size of dataset**: $m = 200$ with 2 classes with size ≈ 100 each.
- **Train-test split**: of 80% and 20%.





Wisconsin Breast Cancer Dataset

- Performance on the Breast Cancer Wisconsin dataset.

Method	d_1	Acc. (%)	Opt. Gap (%)
BDNN	25	69.3	0.0
BDNN_0	25	83.6	0.0
BDNN heur.	25	**95.0**	0.0
BDNN_0 heur.	25	30.0	0.0
DNN	25	91.4	-
BDNN	50	69.3	0.0
BDNN_0	50	84.3	0.51
BDNN heur.	50	89.3	0.0
BDNN_0 heur.	50	71.4	0.0
DNN	50	**91.4**	-

- Accuracy over 10 random shuffles of the data.

Method	d_1	Avg. (%)	Max (%)	Min (%)
BDNN heur.	25	**93.2**	**97.1**	85.0
DNN	25	89.1	91.4	**85.7**

Dataset:

- $N = 9$
- $m = 699$
- 2 classes (**M**alignant & **B**enign)

BDNN:

$$\sigma^k(\alpha) = \begin{cases} \beta_k & \text{if } \alpha < \lambda_k \\ \gamma_k & \text{otherwise} \end{cases}$$

- BDNN
  – λ_k learned
- BDNN_0
  – $\lambda_k = 0$

Conclusion

Conclusion

Results

- Mixed-integer programming formulation to train BDNN
- Heuristic variant has high accuracy on real dataset
- Robust optimization model to enforce robustness during training

Open Problems

- Derivation of more tractable reformulations. (Get rid of the Big-M constraints!)
- Use more general discrete activation functions to increase complexity of the network.
- Can integer programming formulations be used to understand expressivity of neural networks?

For more details:

B. Bah and J. Kurtz, *An Integer Programming Approach to Deep Neural Networks with Binary Activation Functions*, **ICML 2020** Workshop on "Beyond First Order Methods in Machine Learning"

THANK YOU