Training LNNs by Prime-Dual Methods for NSR

Authors: Songtao Lu, Naweed Khan, Ismail Yunus Akhalwaya,
Ryan Riegel, Lior Horesh, Alexander Gray

2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)

Presentation by:

Yao-Ting Chen

Rohan Reddy Sambidi

Joshua Martin Noronha

14 September 2022

Overview

- LNN
- Non-convex constraint problems in LNN
- Algorithms Design
    - ALM
    - ADMM
    - iADMM
- Theoretical Assumptions
- Analysis and Results

1. Logical Neural Network

1. Logical Neural Network

LNN includes two phases: Inference and Training.

- Inference:
    - Evaluation of the activation function for the lower and upper bounds.
    - Upward and downward passes through each neuron, while monotonically tightening the bounds until convergence.

1. Logical Neural Network (contd.)

- Training/Learning:

  Loss function:

  - Supervised labels are provided for truth bounds on a node in the that gives the standard mismatch loss at a neuron.
  - Unsupervised loss that accounts the contradictions at each neurons.
  - Loss is evaluated by $\min_{\mathbf{w}\in\mathscr{W},\ \alpha\in\mathscr{A}} l(\mathbf{w},\alpha)+\tau\psi(\mathbf{w},\alpha)$

    where $\mathbf{w}$ denotes the weights, α the truth threshold meta-parameter, τ is a regularizing meta-parameter, and $\mathscr{W}$, $\mathscr{A}$ represent the feasible sets for $\mathbf{w}$ and α.

1. Logical Neural Network (contd.)

Neuron's parameters $(\mathbf{w}, \beta)$; meta-parameter (α)

Truth table for a binary conjunction neuron

p	q	$p \wedge q$
$p \leq x_F$	$q \leq x_F$	$p \wedge q \leq y_F$
$p \leq x_F$	$q \geq x_T$	$p \wedge q \leq y_F$
$p \geq x_T$	$q \leq x_F$	$p \wedge q \leq y_F$
$p \geq x_T$	$q \geq x_T$	$p \wedge q \geq y_T$

- p and q are real valued inputs
- $x_F = 1 - \alpha$
- $x_T = \alpha$

The activation function, ϕ, is defined by

$$p \wedge q = \phi(\beta - w_p(1 - p) - w_q(1 - q)),$$

with $\mathbf{w} \geq 0 \qquad \beta \geq 0 \qquad 0.5 < \alpha \leq 1$

1.1 LNN- Problem Formulation

The LNN training problem can be formulated with the weighted Łukasiewicz activation function

$$\begin{aligned} \mathbb{P}1 \quad & \min_{\boldsymbol{x}_i \in \mathcal{X}, y_i \in \mathcal{Y}} \quad \frac{1}{n} \sum_{i=1}^{n} l_i(\boldsymbol{x}_i, y_i) + \tau \psi_i(\boldsymbol{x}_i, y_i) \\ \text{s.t.} \quad & x_{i1} - x_{ij} y_i + y_i - 1 - s \leq 0, \forall j \\ & x_{i1} - (1 - y_i) \sum_{j \neq 1} x_{ij} \geq y_i, \end{aligned}$$

where,

l_i , ψ_i denote the loss functions at each neuron,

n is the number of neurons,

$x_i = [\beta; \mathbf{w}_i]$ is a concatenation of the bias and weights that need to be learned at the i^{th} neuron,

$y_i = [\alpha]$ stands for the meta-parameters,

x_{ij} denotes the j^{th} entry of weight at the i^{th} neuron, and

$s > 0$ is some scalar.

1.1 LNN- Problem Formulation (contd.)

We have the following general non-convex constrained problem to solve for:

$$\mathbb{P}2 \qquad \min_{\boldsymbol{x}\in\mathcal{X},\boldsymbol{y}\in\mathcal{Y}} \quad f(\boldsymbol{x},\boldsymbol{y}) \quad \text{s.t.} \quad g(\boldsymbol{x},\boldsymbol{y}) \leq 0$$

where $g(\boldsymbol{x},\boldsymbol{y}) \quad = \quad [g_1(\boldsymbol{x},\boldsymbol{y}),\ldots,g_m(\boldsymbol{x},\boldsymbol{y})]^T$

are non-convex functions, and m denotes the total number of constraints at each neuron.

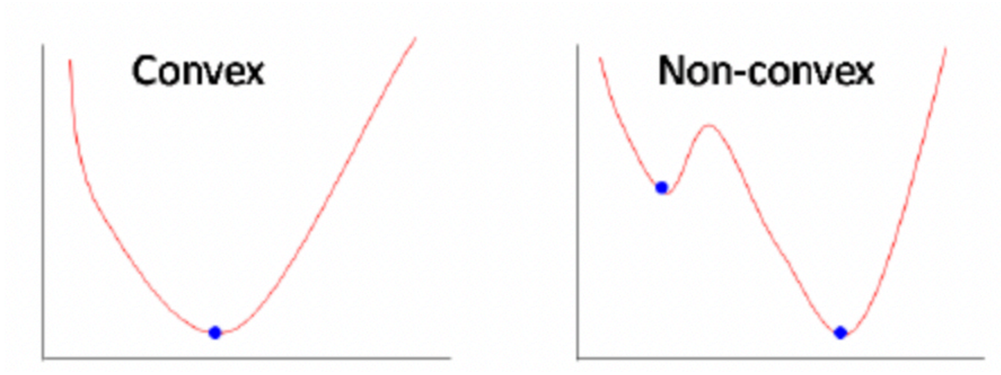
We can write the augmented Lagrangian (AL) of the above problem as

$$\mathcal{L}(\boldsymbol{x},\boldsymbol{y},\boldsymbol{\lambda}) = f(\boldsymbol{x},\boldsymbol{y}) + \frac{\mu}{2}\left\|\left[g(\boldsymbol{x},\boldsymbol{y}) + \frac{\boldsymbol{\lambda}}{\mu}\right]_+\right\|^2 - \frac{\|\boldsymbol{\lambda}\|^2}{2\mu}$$

2. Non-convex Constraint Problems in LNN

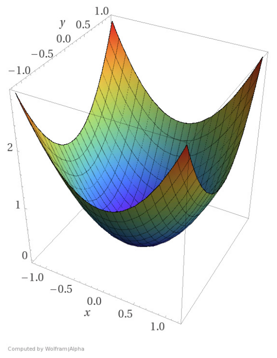
2. Non-convex Constraint Problems in LNN

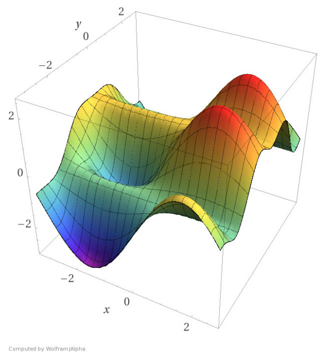
- Real world data contains logical inconsistencies due to which the original LNN does not perform well.
- Humans have been familiar with convex optimization for many years compared to the few contexts where they had to deal with non-convex optimization.
- However, over the last decade, non-convex optimization became more crucial and important than before.
- In fact, with the emergence of deep learning, researchers needed to deal with non-convex optimization more and more given the benefits hidden behind its complexity.



2.1 Convex Optimization Problem

- Convex optimization is a subfield of mathematical optimization that deals with minimizing specific convex function over convex sets.
- It is interesting since in many cases, convergence time is polynomial.
- Linear programming and least square problem are special cases of convex optimization.
- If we can formulate a problem as a convex optimization problem, then we can solve it efficiently, just as we can solve a least square problem efficiently.





2.2 Non-convex Optimization Problem

- A non-convex optimization is any problem where the objective or any of the constraints are non-convex.
- Even simple looking problems with as few as ten variables can be extremely challenging, while problems with a few hundreds of variables can be intractable.
- Here, generally, we face the obligation to compromise, for example - giving up seeking the optimal solution, which minimizes the objective over all feasible points.
- This compromise opens a door for local optimization where intensive research has been conducted and many developments were achieved.
- As a result, local optimization methods are widely used in applications where there is value in finding a good point, if not the very best.

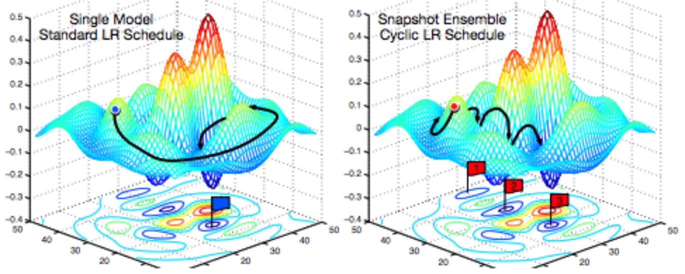
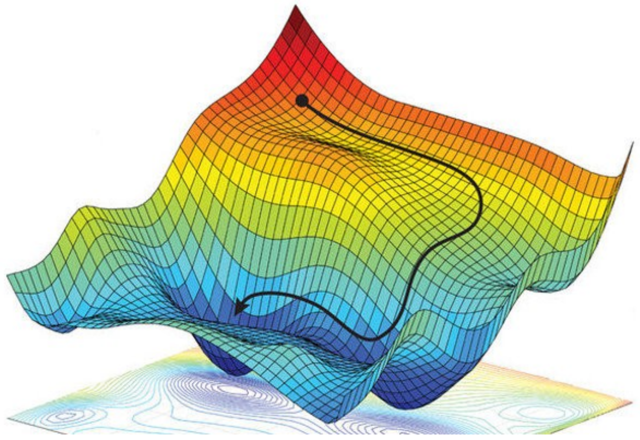


Figure : **Left:** Illustration of SGD optimization with a typical learning rate schedule. The model converges to a minimum at the end of training. **Right:** Illustration of Snapshot Ensembling. The model undergoes several learning rate annealing cycles, converging to and escaping from multiple local minima. We take a snapshot at each minimum for test-time ensembling.

2.3 Why is non-convex optimization needed?

- Non-convex optimization existed since the early days of operations research. Still, the topic gained more interest and focus with the emergence of DL in the recent years. In fact, neural networks are universal function approximatons.
- With enough neurons, they have the ability to approximate any function well.
- Among the functions to be approximated, non-convex functions are gaining more focus. To approximate them, convex functions cannot be good enough. Hence, the importance of using non-convex optimization.
- The freedom to express the learning problem as a non-convex optimization problem gives immense modeling power to the algorithm designer.
- Despite the guarantees achieved in individual instances, it is still complex to unify a framework of what makes non-convex optimization easy to control.



2.4 Non-convex Optimization Convergence

- For non-convex optimization, many convex optimization techniques can be used such as stochastic gradient descent (SGD), mini-batching, stochastic variance-reduced gradient (SVRG), and momentum. There are also specialized methods for solving non-convex problems known in operations research such as alternating minimization methods, branch-and-bound methods. But, these methods are not, generally, very popular for ML problems. In the same context, there are varieties of theoretical convergence results including convergence to a stationary point, convergence to a local minimum, local convergence to the global minimum, and global convergence to the global minimum.

- Non-convex SGD converges with a slow theoretical rate, but not necessarily to a local minimum, which certainly means it doesn’t necessarily reach the global optimum. These theoretical insights can be strengthened through the incorporation of stronger conditions and assumptions. With the previous ones, we can prove its convergence to a local minimum with (or without) an explicit rate of convergence. Sometimes, if we start close enough to the global optimum, we can achieve a local convergence to the global optimum. However, it is very expensive in terms of time and applied to specific cases. Beyond that, the global convergence to the global minimum happens when we can converge wherever we initialize.

3. Algorithms Design

3.1 Prime-Dual Method

- In mathematical optimization theory, duality or the duality principle is the principle that optimization problems may be viewed from either of two perspectives, the primal problem or the dual problem. If the primal is a minimization problem then the dual is a maximization problem (and vice versa). Any feasible solution to the primal (minimization) problem is at least as large as any feasible solution to the dual (maximization) problem. Therefore, the solution to the primal is an upper bound to the solution of the dual, and the solution of the dual is a lower bound to the solution of the primal. In general, the optimal values of the primal and dual problems need not be equal. Their difference is called the duality gap.
- Strong Dual: For convex optimization problems, the duality gap is zero under a constraint qualification condition.
- Weak Dual: duality gap is greater than zero.

$$\min c^{\mathbb{T}} x \quad s.t. \quad Ax \geq b \quad x \geq 0 \qquad\qquad \max b^{\mathbb{T}} y \quad s.t. \quad A^T y \leq c \quad y \geq 0$$

3.2 Duality of convex objective function

Given a minimization problem, $\min_{x \in \mathbb{R}^n} \; f(x),$

$$\text{subject to} \quad h_i(x) \leq 0, \quad i = 1,\ldots,m$$
$$l_j(x) = 0, \quad j = 1,\ldots,r$$

we define the Lagrangian,

$$L(x,u,v) = f(x) + \sum_{i=1}^{m} u_i h_i(x) + \sum_{j=1}^{r} v_j l_j(x)$$

and the Lagrange dual function,

$$g(u,v) = \min_{x \in \mathbb{R}^n} L(x,u,v)$$

The subsequent dual problem is:

$$\max_{u \in \mathbb{R}^m, v \in \mathbb{R}^r} \; g(u,v) \quad \text{subject to} \quad u, v \geq 0$$

3.2 Duality of convex objective function (contd.)

Important properties:

- Dual problem is always convex, i.e., g is always concave (even if primal problem is not convex)
- The primal and dual optimal values, $f*$ and $g*$, always satisfy weak duality: $f* \geq g*$
- Slater's condition: for convex primal, if there is an x such that:

$$h_1(x) < 0, \ldots h_m(x) < 0 \quad \text{and} \quad l_1(x) = 0, \ldots l_r(x) = 0$$

  then strong duality holds: $f* = g*$

Given primal feasible x and dual feasible u, v, the quantity $f(x) - g(u, v)$ is called the duality gap between x and u, v. Note that $f(x) - f* \leq f(x) - g(u, v)$

So, if the duality gap is zero, then x is primal optimal (and similarly, u, v are dual optimal).

From an algorithmic viewpoint, provides a stopping criterion:

if $f(x) - g(u, v) \leq \epsilon$, then we are guaranteed that $f(x) - f* \leq \epsilon$

3.3 Complementary Slackness Condition

If x primal feasible and y dual feasible, then x primal optimal and y dual optimal iff

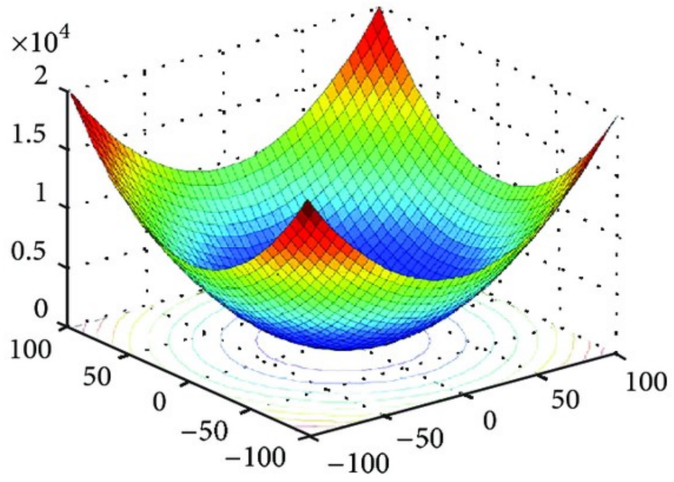
$x_j(y^T A_j - c_j) = 0 \quad \forall j$, and

$y_i(b_i - A_i x) = 0 \quad \forall i$.

If X_0 and Y_0 are the optimal solution of both primal and dual, then we can conclude that $CX_0 = b^T Y_0$

Primal: $\max Z = CX$ s.t. $\begin{cases} AX \leq b \\ X \geq 0 \end{cases}$

Dual: $\min W = b^T Y$ s.t. $\begin{cases} A^T Y \geq C^T \\ Y \geq 0 \end{cases}$



3.4 Lagrange Multipliers

- In mathematical optimization, the method of Lagrange multipliers is a strategy for finding the local maxima and minima of a function subject to equality.
- The basic idea is to convert a constrained problem into a form such that the derivative test of an unconstrained problem can still be applied.
- The relationship between the gradient of the function and gradients of the constraints rather naturally leads to a reformulation of the original problem, known as the Lagrangian function.

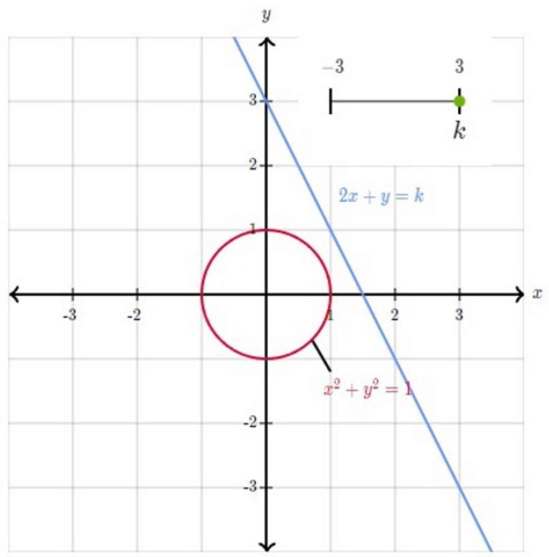
$$L(x_1, x_2, \lambda) = f(x_1, x_2) + \lambda g(x_1 \, . \, x_2)$$

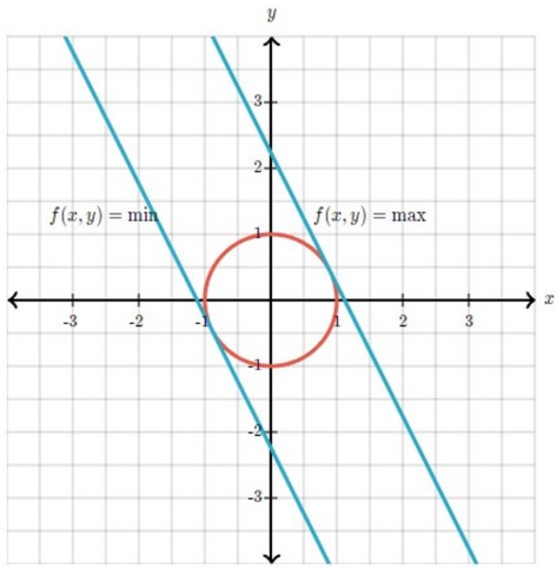
- Lagrange wrote down a special new function which takes in all the same input variables as f and g, along with the new λ, thought of now as a variable rather than a constant.

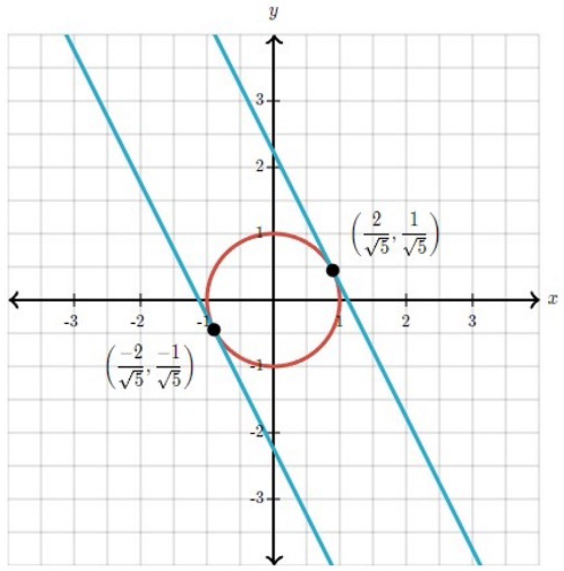
$$\frac{\partial L}{\partial x_1} = 0, \frac{\partial L}{\partial x_2} = 0, \frac{\partial L}{\partial \lambda} = 0$$

- Set the gradient of L equal to the zero vector. In other words, find the critical points of L.

Example







$$\nabla L = \begin{bmatrix} \dfrac{\delta(2x + y - \lambda(x^2 + y^2 - 1))}{\delta x} \\ \dfrac{\delta(2x + y - \lambda(x^2 + y^2 - 1))}{\delta y} \\ \dfrac{\delta(2x + y - \lambda(x^2 + y^2 - 1))}{\delta \lambda} \end{bmatrix} = \begin{bmatrix} 2 - 2\lambda x \\ 1 - 2\lambda y \\ -x^2 - y^2 + 1 \end{bmatrix}$$

3.5 Karush-Kuhn-Tucker Condition (saddle point theorem)

- In the convex and non-convex optimization theory, Karush-Kuhn-Tucker condition is the necessary condition for a constrained objective function to have the most optimized solution.
- Allowing inequality constraints, the KKT approach to nonlinear programming generalizes the method of Lagrange multipliers, which allows only equality constraints.
- Similar to the Lagrange approach, the constrained maximization (minimization) problem is rewritten as a Lagrange function whose optimal point is a saddle point, i.e. a global maximum (minimum) over the domain of the choice variables and a global minimum (maximum) over the multipliers, which is why the Karush–Kuhn–Tucker theorem is sometimes referred to as the saddle-point theorem.

$$\min_{x\in\mathbb{R}^n} \ f(x)$$

subject to $h_i(x) \le 0, \quad i = 1,...,m$

$l_j(x) = 0, \quad j = 1,...,r$

$$0 \in \partial f(x) + \sum_{i=1}^{m} u_i \partial h_i(x) + \sum_{j=1}^{r} v_j \partial l_j(x)$$ (stationary)

$$u_i \cdot h_i(x) = 0, \ \forall i$$ (complementary slackness)

$$h_i \le 0, \quad l_i(x) = 0, \ \forall i, j$$ (primal feasibility)

$$u_i \ge 0, \ \forall i$$ (dual feasibility)

3.5 Karush-Kuhn-Tucker Condition (saddle point theorem) (contd.)

- Let x^* and u^*, v^* be primal and dual solutions with zero duality gap (strong duality holds). Then,

$$\begin{aligned} f(x^*) &= g(u^*, v^*) \\ &= min_{x \in \mathbb{R}} f(x) + \sum_{i=1}^{m} u_i^* h_i(x) + \sum_{j=1}^{r} v_j^* l_j(x) \\ &\leq f(x^*) + \sum_{i=1}^{m} u_i^* h_i(x) + \sum_{j=1}^{r} v_j^* l_j(x) \\ &\leq f(x^*) \end{aligned}$$

- Two things are important:
  - The point x^* minimizes $L(x, u^*, v^*)$ over $x \in \mathbb{R}^n$. Hence the sub-differential of $L(x, u^*, v^*)$ must contain 0 at $x = x^*$ (this is exactly the stationarity condition).
  - We must have $\sum_{i=1}^{m} u_i^* h_i(x^*) = 0$, and since each term here is ≤ 0, this implies $u_i^* h_i(x^*) = 0 \quad \forall i$ (this is exactly complementary slackness).

- Conclusion: If x^* and u^*, v^* are primal and dual solutions, with zero duality gap, then x^*, u^*, v^* satisfy the KKT conditions.

3.6 Dual Ascent Method

- Dual Gradient Descent is a popular method for optimizing an objective under a constraint. In reinforcement learning, it helps us to make better decisions.

$$\min_x f(x) \quad \text{such that} \quad C(x) = 0$$

- The key idea is transforming the objective into a Lagrange dual function which can be optimized iteratively.

$$L(x,\lambda) = f(x) + \lambda C(x)$$

$$g(\lambda) = L(x^*(\lambda),\lambda) \quad \text{where} \quad x^* \quad \arg\min_x L(x,\lambda)$$

where λ is a scalar which we call the Lagrangian multiplier.

- Augmented Lagrangian:

$$L_c(x,\lambda) = f(x) + \lambda^T h(x) + \frac{c}{2}\|h(x)\|^2$$

3.6 Dual Ascent Method (contd.)

- The dual function L is a lower bound for the original optimization problem. Indeed, if the function f is a convex function, the strong duality will often hold which say the maximum value of g equals the minimum values of the optimization problem. Hence, if we find λ that maximize g, we solve the optimization problem.

Step-1
$$\sum_{i=1}^{m} u_i^* h_i(x^*) = 0$$

Step-2
$$L(x, u, v) = f(x) + \sum_{i=1}^{m} u_i h_i(x) + \sum_{j=1}^{r} v_j l_j(x)$$

Step-3
$$g(u, v) = \min_{x \in \mathbb{R}^n} L(x, u, v)$$

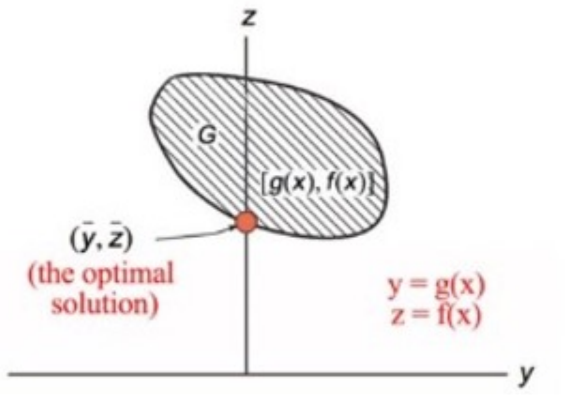
Example

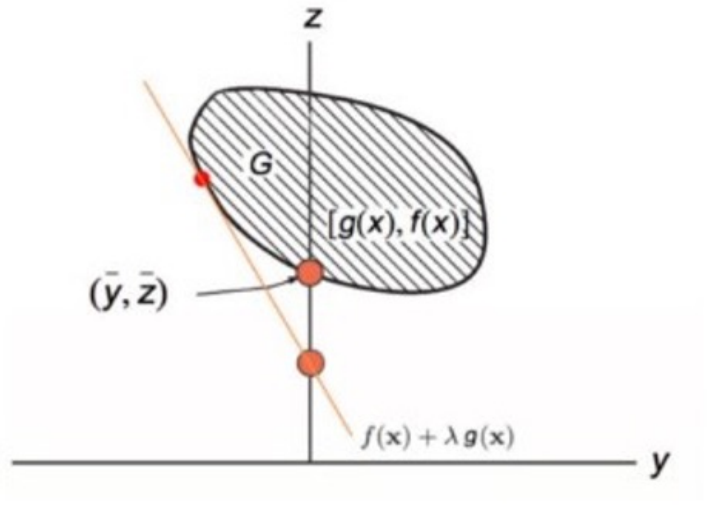
Primal Problem P

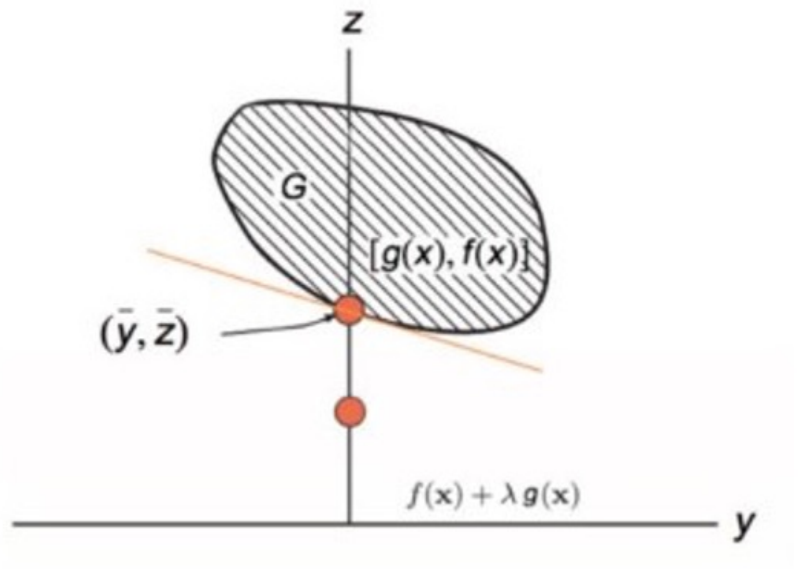
minimise $f(x)$,

subject to:

$g(x) \leq 0$,







3.7 Penalty function

- Penalty methods are a certain class of algorithms for solving constrained optimization problems.
- A penalty method replaces a constrained optimization problem by a series of unconstrained problems whose solutions ideally converge to the solution of the original constrained problem.
- The unconstrained problems are formed by adding a term, called a penalty function, to the objective function that consists of a penalty parameter multiplied by a measure of violation of the constraints.
- The measure of violation is nonzero when the constraints are violated and is zero in the region where constraints are not violated.

Example

- We are solving the following constrained problem:

$$\min f(x)$$

$$\text{s.t. } c_i(x) \leq 0, \forall i \in I$$

- This problem can be solved as a series of unconstrained minimization problems

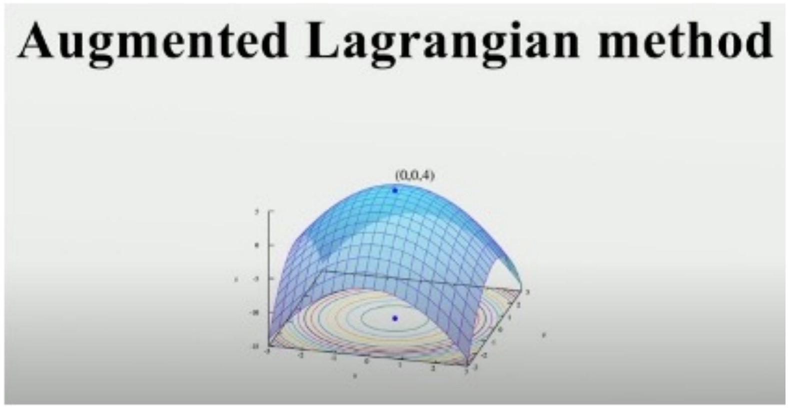
$$\min \Phi_k(x) = f(x) + \sigma_k \sum_{i \in I} g(c_i(x))$$

Where, $g(c_i(x)) = \max(0, c_i(x))^2$

- In the above equations, $g(c_i(x))$ is the exterior penalty function while σ_k are the penalty coefficients.
- In each iteration k of the method, we increase the penalty coefficient σ_k (e.g. by a factor of 10), solve the unconstrained problem and use the solution as the initial guess for the next iteration.
- Solutions of the successive unconstrained problems will eventually converge to the solution of the original constrained problem.

3.8 Augmented Lagrangian Method

- Augmented Lagrangian methods are a certain class of algorithms for solving constrained optimization problems.
- They have similarities to penalty methods in that they replace a constrained optimization problem by a series of unconstrained problems and add a penalty term to the objective; the difference is that the augmented Lagrangian method adds yet another term, designed to mimic a Lagrange multiplier.



3.8 Augmented Lagrangian Method (contd.)

minimize $f(x)$

subject to $x \in X, \quad \|Ax - b\| = 0,$

where $f : \mathbb{R}^n \to \mathbb{R}$ and $h : \mathbb{R}^n \to \mathbb{R}^m$ are continuous, and X is closed.

- Convergence of dual methods can be greatly improved by utilizing augmented Lagrangian. Start by transforming primal.

$$x \in \mathbb{R}^n f(x) + \frac{\rho}{2} \| Ax - b \|^2 \qquad s.t. \; Ax = b$$

- The augmented Lagrangian is (with $\rho > 0$):

$$L(x, \lambda; \rho) := \underbrace{f(x) + \lambda^T (Ax - b)}_{\text{Lagrangian}} + \underbrace{\frac{\rho}{2} \| Ax - b \|_2^2}_{\text{Augmentation}}$$

$$\lambda^{i+1} = \lambda_i + \mu_k c_i(x_k)$$

$$x_k = \arg\min \Phi_k(x)$$

3.9 iADMM

Algorithm 1 iADMM

```
**Initialization:** $\sigma > 1$, $\boldsymbol{\lambda}_0 = 0$, $\alpha_0$, $\beta_0$, $\gamma_0$
**for** $r = 1, \dots, T$ **do**
    Let $\alpha_r = \alpha_0 \sigma^r$ and $\boldsymbol{\theta}_0 = \boldsymbol{x}_r$.
    **for** $t = 1, \dots$ **do** Update $\boldsymbol{\theta}_t$ by (8).                ▷ $\boldsymbol{x}$-inner loop
        **if** $2\rho\|\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t\| \le \epsilon/4$ **then** return $\boldsymbol{x}_{r+1} = \boldsymbol{\theta}_{t+1}$
        **end if**
    **end for**
    Let $\beta_r = \beta_0 \sigma^r$ and $\boldsymbol{\vartheta}_0 = \boldsymbol{y}_r$.
    **for** $t = 1, \dots$ **do** Update $\boldsymbol{\vartheta}_t$ by (9)                ▷ $\boldsymbol{y}$-inner loop
        **if** $2\rho\|\boldsymbol{\vartheta}_{t+1} - \boldsymbol{\vartheta}_t\| \le \epsilon/4$ **then** return $\boldsymbol{y}_{r+1} = \boldsymbol{\vartheta}_{t+1}$
        **end if**
    **end for**
    Update $\boldsymbol{\lambda}_{r+1}$ by (10)
**end for**
```

3.9 iADMM (contd.)

$$\min_{\boldsymbol{x}\in\mathscr{X},\, \boldsymbol{y}\in\mathscr{Y}} \max_{\boldsymbol{\lambda}\geq 0} \mathscr{L}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\lambda})$$

Step-1

$$\boldsymbol{\theta}_{t+1} = \arg\min_{\boldsymbol{\theta}\in\mathscr{X}} \boldsymbol{\mathscr{L}}(\boldsymbol{\theta}, \boldsymbol{y}_r, \boldsymbol{\lambda}_r) + \frac{\alpha_r}{2}\|\boldsymbol{\theta} - \boldsymbol{\theta}_t\|^2$$

Step-2

$$\boldsymbol{\vartheta}_{t+1} = \arg\min_{\boldsymbol{\vartheta}\in\mathscr{Y}} \boldsymbol{\mathscr{L}}(\boldsymbol{x}_{r+1}, \boldsymbol{\vartheta}, \boldsymbol{\lambda}_r) + \frac{\beta_r}{2}\|\boldsymbol{\vartheta} - \boldsymbol{\vartheta}_t\|^2$$

Step-3

$$\boldsymbol{\lambda}_{r+1} = [\boldsymbol{\lambda}_r + \gamma_{r+1}\; g(\boldsymbol{x}_{r+1}, \boldsymbol{y}_{r+1})]_+$$

4. Theoretical Assumptions

4. Assumptions

A1. Function f is smooth and weakly convex with respect to $\boldsymbol{x}$ and $\boldsymbol{y}$ respectively with constant ρ, i.e., $f + \frac{\rho}{2}\|\cdot\|^2$ is convex.

A2. Function g is Lipschitz continuous *w.r.t.* each block.

A3. The Jacobian matrices $J_g(\boldsymbol{x}, \boldsymbol{y})$ and $J'_g(\boldsymbol{x}, \boldsymbol{y})$ of function g *w.r.t.* block $\boldsymbol{x}$ and $\boldsymbol{y}$ are also Lipschitz continuous, i.e., $\|J_g(\boldsymbol{x}, \boldsymbol{y}) - J_g(\boldsymbol{x}', \boldsymbol{y})\| \leq G\|\boldsymbol{x} - \boldsymbol{x}'\|$ and $\|J'_g(\boldsymbol{x}, \boldsymbol{y}) - J'_g(\boldsymbol{x}, \boldsymbol{y}')\| \leq G\|\boldsymbol{y} - \boldsymbol{y}'\|$.

A4. Each entry of $\boldsymbol{y}$ is lower bounded above zero, i.e., $\boldsymbol{y} \geq \kappa$.

A5. f is lower bounded, and $\mathcal{X}, \mathcal{Y}$ are convex and compact.

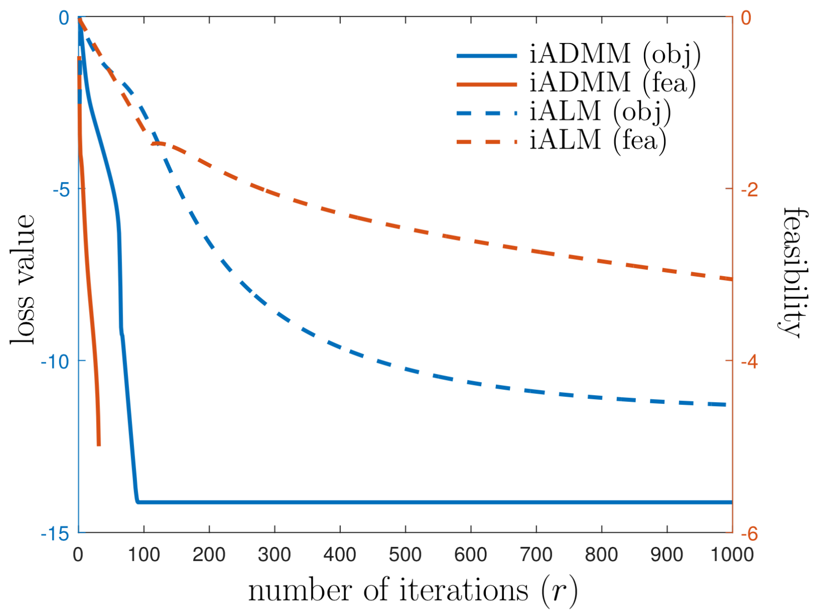
Analysis and Results

Toy Example

$$f(x, y) = \left\| x^{\Phi 2} y - 2 \cdot 1 \right\|$$

where $x^{\Phi 2}$ = component wise squared of x.

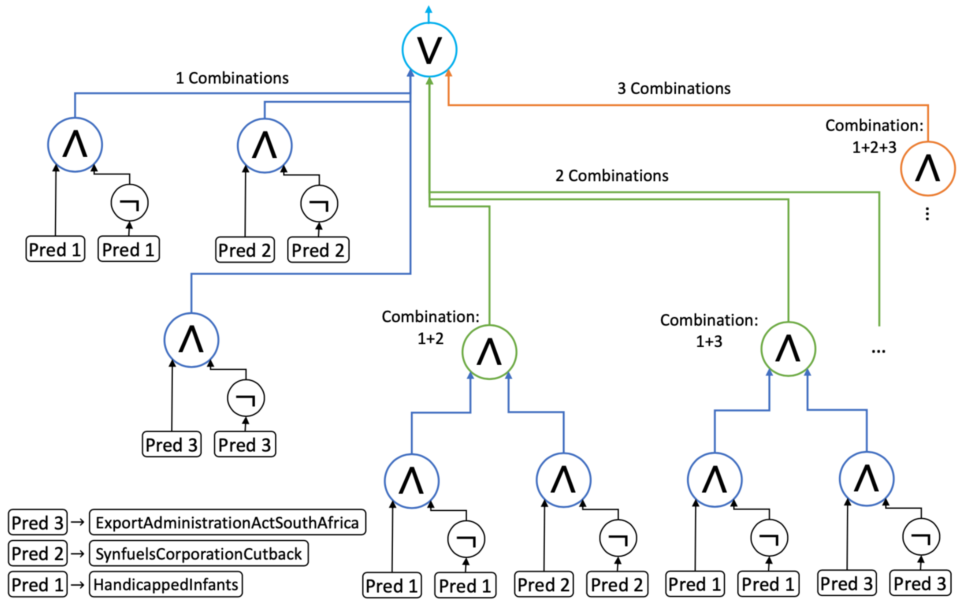
Constraints: $x^T y - 2 \leq 0$ and $1 - x^T y \leq 0$



- For a large inner loop step size, the iALM algorithm will diverge
- iADMM converges much faster than iALM

Real World Problem Analysis: Congressional Voting Records Data Set

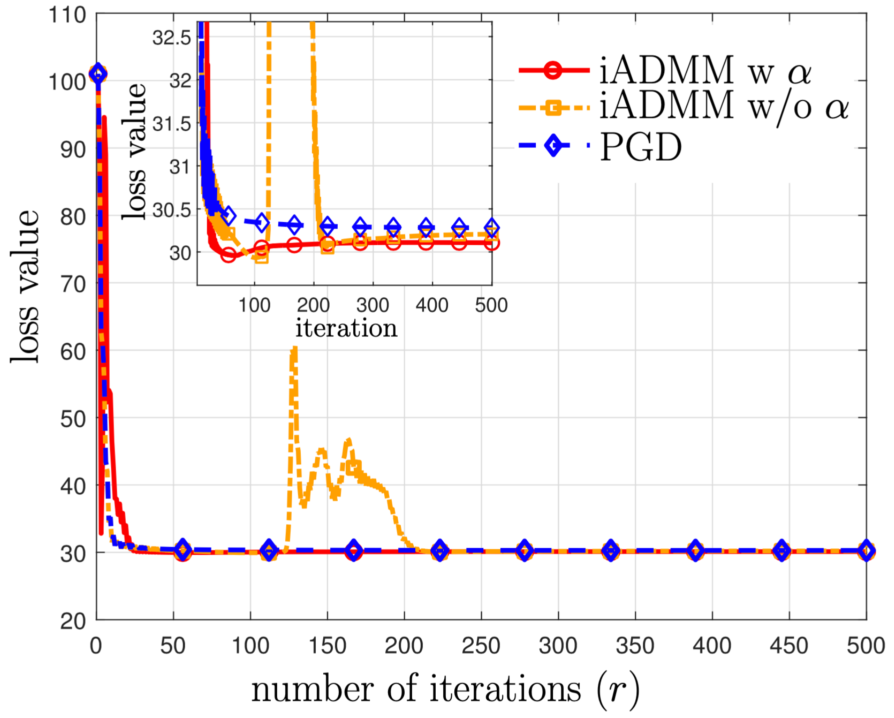
- The dataset contains 435 data points and 16 attributes
- Test to see if the network learns the weights on its own
- PGD => Projected Gradient Descent (blue)
- Two types of iADMM methods are used:
    - iADMM where α is learnable (orange)
    - iADMM where α is not learnable (yellow)



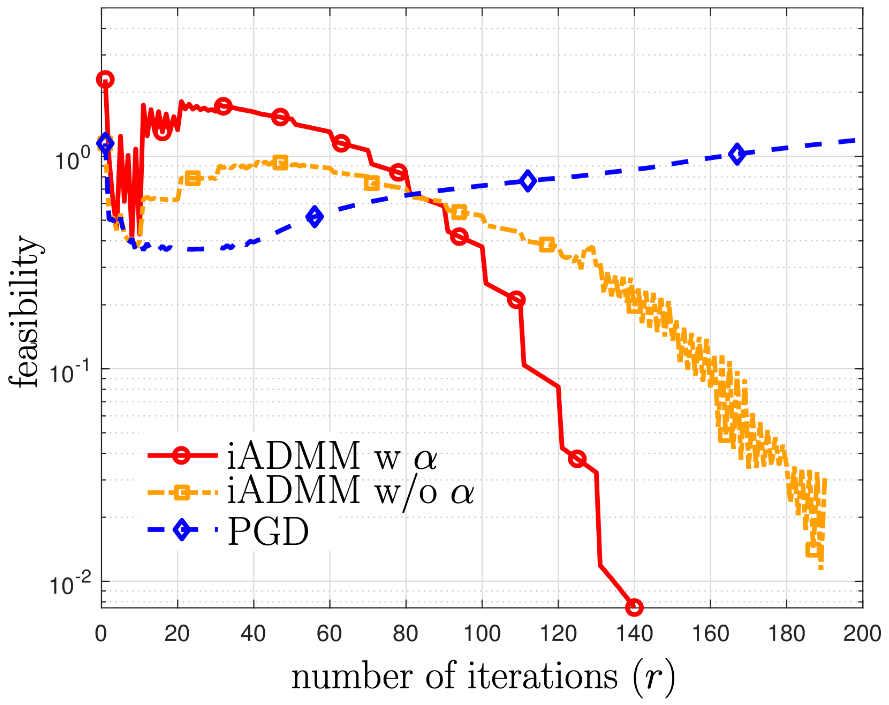
Ref: https://archive.ics.uci.edu/ml/datasets/congressional+voting+records

Real World Problem Analysis: Congressional Voting Records Data Set (contd.)

Loss value of the LNN



Feasibility error



Takeaways

- The LNN model with the learnable α parameter performed better than the rest for this particular problem with minimal constraint violation
- iADMM shows some oscillation during the process of convergence due to the nature of the min-max game
- The solution with PGD had logical inconsistencies
- Even though all models had approximately similar loss values, the iADMM model had more feasibility in its solution

References

1. S. Lu, N. Khan, I. Y. Akhalwaya, R. Riegel, L. Horesh, and A. Gray, “Training Logical Neural Networks By Primal–Dual Methods For Neuro-Symbolic Reasoning,” IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Jun. 2021.
2. R. Riegel et al., “Logical Neural Networks,” NeurIPS, Jun. 2020.
3. M. Zibulevsky, “Penalty Function and Augmented Lagrangian Methods,” Nov. 2013. [Video] .Youtube. https://www.youtube.com/watch?v=9jcWM-TUQHk
4. Center for Science of Information NSF STC. (Jan 25 2021). Logical Neural Networks: Toward Unifying Statistical and Symbolic AI by Alexander Gray. [Video]. Youtube. https://www.youtube.com/watch?v=kAaKs_04dEk
5. Aarti Singh, Pradeep Ravikumar. Augmented Lagrangian & the Method of Multipliers [slides]. https://www.cs.cmu.edu/~pradeepr/convexopt/Lecture_Slides/Augmented-lagrangian.pdf
6. M. Zibulevsky, “Penalty Multiplier Method (Augmented Lagrangian) 1,” Sept. 2016. [video]. Youtube. https://www.youtube.com/watch?v=IVIJSHHkqPo&ab_channel=MichaelZibulevsky

Ira A. Fulton Schools of Engineering

Arizona State University