Include Optimization as a Layer in NN

Outline

- What is $\boxed{\text{OptNet}}$ / feedforward pass $\Rightarrow$
- How to update parameters in the OptNet / backward pass $\Leftarrow$
- Application using OptNet
  - Representation power
  - MINIST $\leftarrow$ OptNet
  - Sodoku $\leftarrow$ 〃

implicit differentiation.

OptNet: Differentiable Optimization as a Layer in Neural Networks

What is OptNet

- **Feedforward NN**

Form $\underline{z_{i+1}} = \underline{\sigma(W_i z_i + b_i)}$

Parameters $\underline{W_i}, \underline{b_i}$

- **OptNet**

quadractic objective

$$
\begin{aligned}
z_{i+1} = \operatorname*{argmin}_{z} \quad & \frac{1}{2} z^T Q(z_i) z + q(z_i)^T z \\
\text{subject to} \quad & A(z_i) z = b(z_i) \Rightarrow \text{等式} \\
& G(z_i) z \leq h(z_i) \Rightarrow \text{不等式.}
\end{aligned} \tag{1}
$$

$\Rightarrow \underline{Q}(z_i), \underline{q}(z_i), \underline{A}(z_i), \underline{b}(z_i), \underline{G}(z_i), \underline{h}(z_i)$

Can depend on the previous layer z_i

What is OptNet

- **Feedforward NN**
- **OptNet**

Form

$$z_{i+1} = \sigma(W_i z_i + b_i)$$

$$z_{i+1} = \underset{z}{\operatorname{argmin}} \ \frac{1}{2} z^T Q(z_i) z + q(z_i)^T z$$
$$\text{subject to} \ A(z_i) z = b(z_i)$$
$$G(z_i) z \leq h(z_i) \tag{1}$$

Parameters

$$W_i, b_i$$

$$Q(z_i), \boxed{q(z_i), A(z_i), b(z_i), G(z_i), h(z_i)}$$

Gradients

$$\frac{\partial z_{i+1}}{\partial W_i} = \frac{\partial}{\partial W_i}(relu(W_i z_i + b))$$

$$\Leftarrow \quad \frac{\partial z_{i+1}}{\partial Q(z_i)}, \frac{\partial z_{i+1}}{\partial q(z_i)}, \ldots, ?$$

$$\frac{\partial L}{\partial W_i} = \frac{\partial L}{\partial z_{i+1}} \cdot \frac{\partial z_{i+1}}{\partial W_i}$$

$$\frac{\partial z_{i+1}}{\partial b_i}$$

$$= diag(relu(sign(b + W_i z_i)))_i^T \otimes z_i$$

How to update parameters in the OptNet

$$
\begin{aligned}
z_{i+1} = \underset{z}{\operatorname{argmin}} \quad & \frac{1}{2} z^T Q(z_i) z + q(z_i)^T z \\
\text{subject to} \quad & A(z_i) z = b(z_i) \\
& G(z_i) z \leq h(z_i)
\end{aligned}
\tag{1}
$$

- Output of $i, i+1$ layer, Variable $\quad \underline{z_i, z_{i+1} : \mathbb{R}^n}$
- Problem data of optimization problem, Parameter in NN

$$\underline{Q \in \mathbb{R}^{n\times n}}, \underline{q \in \mathbb{R}^n}, \underline{A \in \mathbb{R}^{m\times n}}, \underbrace{b \in \mathbb{R}^m}_{\Downarrow}, \underbrace{G \in \mathbb{R}^{p\times n}, h \in \mathbb{R}^p}_{\Downarrow}$$

m个等式约束 n个不等式

- Gradients: derivative of the solution to its parameter

$$\frac{\partial z_{i+1}}{\partial Q(z_i)}, \frac{\partial z_{i+1}}{\partial q(z_i)}, \ldots, ?$$

How to update parameters in the OptNet

$$\begin{aligned} z_{i+1} = \operatorname*{argmin}_z \quad & \frac{1}{2} z^T Q(z_i) z + q(z_i)^T z \\ \text{subject to} \quad & A(z_i) z = b(z_i) \\ & G(z_i) z \leq h(z_i) \end{aligned} \tag{1}$$

Qz -> dQz+Qdz

- Write Lagrangian $L(z, \nu, \lambda) = \frac{1}{2} z^T Q z + q^T z + \nu^T (Az - b) + \lambda^T (Gz - h)$
- Write KKT conditions

$$\frac{dL}{dz} = 0, \underbrace{\frac{dL}{d\lambda} = 0, \frac{dL}{dv} = 0}_{\text{dual variable}}$$

$$\boxed{\begin{aligned} Qz^\star + q + A^T \nu^\star + G^T \lambda^\star = 0 \\ Az^\star - b = 0 \\ D(\lambda^\star)(Gz^\star - h) = 0, \end{aligned}}$$

1

z^*, v^*, λ^* are the optimal primal and dual variables

$$\boxed{z^* = z_{i+1}}$$

How to update parameters in the OptNet

$$
\begin{aligned}
z_{i+1} = \underset{z}{\operatorname{argmin}} \quad & \frac{1}{2} z^T Q(z_i) z + q(z_i)^T z \\
\text{subject to} \quad & A(z_i) z = b(z_i) \\
& G(z_i) z \le h(z_i)
\end{aligned}
\tag{1}
$$

- Write derivative of KKT conditions

$$
\begin{aligned}
\mathsf{d}Qz^\star + Q\mathsf{d}z + \mathsf{d}q + \mathsf{d}A^T\nu^\star + \\
A^T\mathsf{d}\nu + \mathsf{d}G^T\lambda^\star + G^T\mathsf{d}\lambda = 0 \\
\mathsf{d}Az^\star + A\mathsf{d}z - \mathsf{d}b = 0 \\
D(Gz^\star - h)\mathsf{d}\lambda + D(\lambda^\star)(\mathsf{d}Gz^\star + G\mathsf{d}z - \mathsf{d}h) = 0
\end{aligned}
$$

②

$ab \quad da \cdot b + a \cdot db$

$$\longleftrightarrow \; =$$

$$
\begin{bmatrix} Q & G^T & A^T \\ D(\lambda^\star)G & D(Gz^\star - h) & 0 \\ A & 0 & 0 \end{bmatrix}
\begin{bmatrix} \mathsf{d}z \\ \mathsf{d}\lambda \\ \mathsf{d}\nu \end{bmatrix} =
- \begin{bmatrix} \mathsf{d}Qz^\star + \mathsf{d}q + \mathsf{d}G^T\lambda^\star + \mathsf{d}A^T\nu^\star \\ D(\lambda^\star)\mathsf{d}Gz^\star - D(\lambda^\star)\mathsf{d}h \\ \mathsf{d}Az^\star - \mathsf{d}b \end{bmatrix}.
$$

③

$$\Rightarrow \boxed{\frac{\partial z_{i+1}}{\partial Q(z_i)}}, \frac{\partial z_{i+1}}{\partial q(z_i)}, \ldots, ?$$

$dq \cdot dA \Rightarrow 0.$

How to update parameters in the OptNet

$$
\begin{bmatrix} Q & G^T & A^T \\ D(\lambda^\star)G & D(Gz^\star - h) & 0 \\ A & 0 & 0 \end{bmatrix}
\begin{bmatrix} \mathrm{d}z \\ \mathrm{d}\lambda \\ \mathrm{d}\nu \end{bmatrix} =
- \begin{bmatrix} \mathrm{d}Qz^\star + \mathrm{d}q + \mathrm{d}G^T\lambda^\star + \mathrm{d}A^T\nu^\star \\ D(\lambda^\star)\mathrm{d}Gz^\star - D(\lambda^\star)\mathrm{d}h \\ \mathrm{d}Az^\star - \mathrm{d}b \end{bmatrix}.
$$

$$
\frac{\partial z_{i+1}}{\partial Q(z_i)}, \frac{\partial z_{i+1}}{\partial q(z_i)}, \ldots, ?
$$

$l = c^T x$

标量　$n \times 1$　$n \times 1$

$\frac{dl}{dc}$

$dl = \boxed{dc^T} \cdot x + c^T dx.$　I　$0.$

$\frac{dl}{dc_i} = \frac{dc^T}{dc_i} \cdot x + \cancel{c^T \frac{dx}{dc_i}}.$

$= [0 \cdots 0 \; 1 \; 0 \cdots 0] \, x.$

ith

$\frac{dl}{dc} = \begin{bmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{bmatrix} x.$

How to update parameters in the OptNet

$$\overbrace{\begin{bmatrix} Q & G^T & A^T \\ D(\lambda^\star)G & D(Gz^\star - h) & 0 \\ A & 0 & 0 \end{bmatrix}}^{K} \begin{bmatrix} \mathrm{d}z \\ \mathrm{d}\lambda \\ \mathrm{d}\nu \end{bmatrix} = - \begin{bmatrix} \mathrm{d}\underbrace{Q}_{I}z^\star + \mathrm{d}q + \mathrm{d}G^T\lambda^\star + \mathrm{d}A^T\nu^\star \\ D(\lambda^\star)\mathrm{d}Gz^\star - D(\lambda^\star)\mathrm{d}h \\ \mathrm{d}Az^\star - \mathrm{d}b \end{bmatrix}.$$

dq dG ⇒ 0.

$$\frac{\partial z_{i+1}}{\partial Q(z_i)}, \frac{\partial z_{i+1}}{\partial q(z_i)}, \ldots, ?$$

$$\begin{bmatrix} \frac{dz}{dQ} \\ \frac{d\lambda}{dQ} \\ \frac{dv}{dQ} \end{bmatrix} = -K^{-1} \begin{bmatrix} z^* \otimes I_{n\times n} \\ 0 \\ 0 \end{bmatrix} \qquad \begin{bmatrix} \frac{dz}{dq} \\ \frac{d\lambda}{dq} \\ \frac{dv}{dq} \end{bmatrix} = -K^{-1} \begin{bmatrix} I \\ 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} \frac{dz}{db} \\ \frac{d\lambda}{db} \\ \frac{dv}{db} \end{bmatrix} = -K^{-1} \begin{bmatrix} 0 \\ 0 \\ -I \end{bmatrix}$$

$$\begin{bmatrix} \frac{dz}{dh} \\ \frac{d\lambda}{dh} \\ \frac{dv}{dh} \end{bmatrix} = -K^{-1} \begin{bmatrix} 0 \\ -D(\lambda^*) \\ 0 \end{bmatrix}$$

How to update parameters in the OptNet

- To include previous backward pass vector $\frac{\partial l}{\partial z^*} \in \mathbb{R}^n$
  - loss
  - We actually want $\nabla_b l = \frac{\partial l}{\partial z^*}\frac{\partial z^*}{\partial b}, \nabla_Q l, \nabla_A l, \ldots$

$$\begin{bmatrix} Q & G^T & A^T \\ D(\lambda^\star)G & D(Gz^\star - h) & 0 \\ A & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathrm{d}z \\ \mathrm{d}\lambda \\ \mathrm{d}\nu \end{bmatrix} = - \begin{bmatrix} \mathrm{d}Qz^\star + \mathrm{d}q + \mathrm{d}G^T\lambda^\star + \mathrm{d}A^T\nu^\star \\ D(\lambda^\star)\mathrm{d}Gz^\star - D(\lambda^\star)\mathrm{d}h \\ \mathrm{d}Az^\star - \mathrm{d}b \end{bmatrix}.$$

$$\frac{\partial l}{\partial b} = \begin{bmatrix} \frac{\partial l}{\partial z} & 0 & 0 \end{bmatrix} \begin{bmatrix} \frac{\partial z}{\partial b} \\ \frac{\partial \lambda}{\partial b} \\ \frac{\partial \nu}{\partial b} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{\partial l}{\partial z} & 0 & 0 \end{bmatrix} K^{-1} \begin{bmatrix} 0 \\ 0 \\ I \end{bmatrix}$$

$$\frac{\partial l}{\partial Q} = [\quad] K^{-T} \cdot \begin{bmatrix} \left(\frac{\partial l}{\partial z}\right)^T \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & I \end{bmatrix} (K^T)^{-1} \cdot \begin{bmatrix} \left(\frac{\partial l}{\partial z}\right)^T \\ 0 \\ 0 \end{bmatrix}$$

How to update parameters in the OptNet

- To include previous backward pass vector $\dfrac{\partial l}{\partial z^*} \in \mathbb{R}^n$
  - We actually want $\nabla_b l = \dfrac{\partial l}{\partial z^*}\dfrac{\partial z^*}{\partial b}, \nabla_Q l, \nabla_A l, \ldots$

$$\begin{bmatrix} Q & G^T & A^T \\ D(\lambda^\star)G & D(Gz^\star - h) & 0 \\ A & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathsf{d}z \\ \mathsf{d}\lambda \\ \mathsf{d}\nu \end{bmatrix} = - \begin{bmatrix} \mathsf{d}Qz^\star + \mathsf{d}q + \mathsf{d}G^T\lambda^\star + \mathsf{d}A^T\nu^\star \\ D(\lambda^\star)\mathsf{d}Gz^\star - D(\lambda^\star)\mathsf{d}h \\ \mathsf{d}Az^\star - \mathsf{d}b \end{bmatrix}.$$

the differential matrix

$(K^T)^{-1}$

$$\begin{bmatrix} d_z \\ d_\lambda \\ d_\nu \end{bmatrix} = - \begin{bmatrix} Q & G^T D(\lambda^\star) & A^T \\ G & D(Gz^\star - h) & 0 \\ A & 0 & 0 \end{bmatrix}^{-1} \begin{bmatrix} \left(\frac{\partial \ell}{\partial z^\star}\right)^T \\ 0 \\ 0 \end{bmatrix} \tag{7}$$

then the relevant gradients with respect to all the QP parameters can be given by

$$\begin{aligned} \nabla_Q \ell &= \frac{1}{2}(d_z z^T + z d_z^T) & \nabla_q \ell &= d_z \\ \nabla_A \ell &= d_\nu z^T + \nu d_z^T & \nabla_b \ell &= -d_\nu \\ \nabla_G \ell &= D(\lambda^\star)(d_\lambda z^T + \lambda d_z^T) & \nabla_h \ell &= -D(\lambda^\star) d_\lambda \end{aligned} \tag{8}$$

Application using OptNet

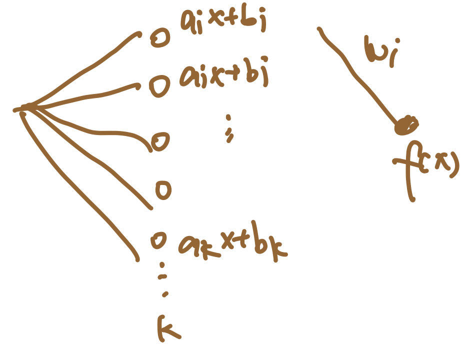
- Representation power: OptNet layer can represent any piecewise linear univariate function

ReLU.

$$f(x) = \sum_{i=1}^{k} w_i \max(a_i x + b, 0)$$

两层神经网络

$$z_{i+1} = \begin{array}{ll} \underset{z\in\mathbb{R}, t\in\mathbb{R}^k}{\text{minimize}} & \|t\|_2^2 + (z - w^T t)^2 \\ \text{subject to} & a_i x + b_i \le t_i, \quad i = 1, \ldots, k \end{array}$$



$t_i = 0$

$t_i = a_i x + b_i \Rightarrow \mathrm{ReLU}(a_i x + b_i)$

$$z = \sum_{i=1}^{k} w_i \max(a_i x + b_i, 0) = f(x).$$

Application using OptNet

- Representation power: converse is False

$$f'(x) = \boxed{\max\{a_1^T x, a_2^T x, a_3^T x\}}$$

ReLU nature $\boxed{\max\{a_i^T x + b_i, 0\}}$ 单层网络无法实现

OptNet

$$\boxed{\begin{aligned} &\underset{z}{\text{minimize}} \quad z^2 \\ &\text{subject to} \quad a_i^T x \le z, \ i = 1, \ldots, 3. \end{aligned}}$$

Application using OptNet

- MINIST:
  - Learn constraints and dependencies over the output or latent space of a model.
  - Show these layers can be included within existing network architectures and efficiently propagate the gradients through the layer.

- Model
  - Fully connected layer: FC600-FC10-FC10-SoftMax
  - Fully connected layer with OptNet: FC600-FC10-Optnet10-SoftMax

$z \in \mathbb{R}^{10}$.

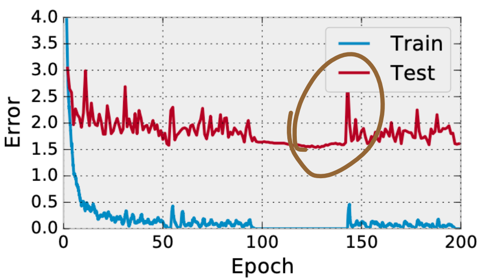
The learnable parameter is L, G, z_0, s_0

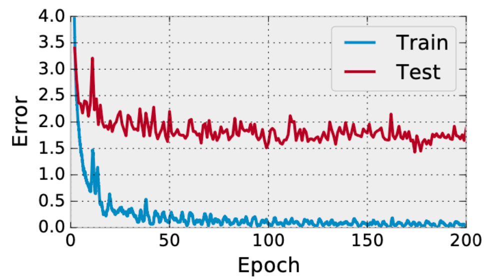
$$\boxed{\begin{aligned} \min_{z \in \mathbb{R}^{10}} \; & z^T Q z + z_i^T z \\ \text{s.t. } & Q = LL^T + \epsilon I, \\ & Gz \le Gz_0 + s_0 \\ & Az = Az_0 \end{aligned}}$$

Application using OptNet

- MINIST:
  - Learn constraints and dependencies over the output or latent space of a model.
  - Show these layers can be included within existing network architectures and efficiently propagate the gradients through the layer.

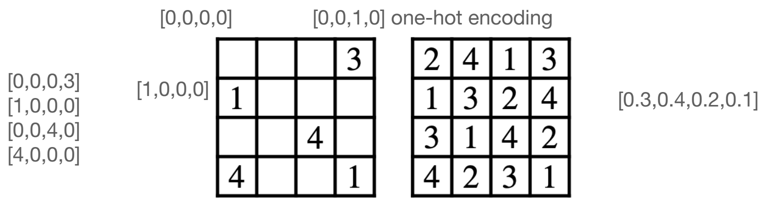
- Result
  - slightly lower error and less variance using the OptNet network





Baseline

Application using OptNet



4x4

- Sudoku
  - a constraint satisfaction problem, nontrivial for computers to know the rules if only given the initial problem and corresponding solved example
  - Show OptNet can capture **complex strict relationships** between all input and output variables
  - Input: $\boxed{4 \times 4} \times 4$ tensor, Output: $\underline{4 \times 4 \times 4}$ tensor (4×4 grid with one-hot encoding)

one-hot encoding

Application using OptNet

- Experiment
  - Train on 9000 puzzles, test on 1000 different puzzles
  - Error rate: the percentage of puzzles solved incorrectly

- Model
  - multilayer feedforward network: 10 convolutional layers with 512 3x3 filters
  - Feedforward NN with OptNet

flat

$B \times 4 \times 4 \times 4$ (Input) $\longrightarrow B \times 64 \longrightarrow$ **OptNet** $\longrightarrow B \times 64 \longrightarrow B \times 4 \times 4 \times 4$ (Output) $\Rightarrow$

grid

$B \times 4 \times 4 \times 4$ (Input) $\longrightarrow B \times 64 \longrightarrow$ **Linear Mapping** $\longrightarrow B \times n \longrightarrow$ **OptNet** $\longrightarrow B \times n \longrightarrow$ **Linear Mapping** $\longrightarrow B \times 64 \longrightarrow B \times 4 \times 4 \times 4$ (Output)

Application using OptNet

- Model
  - multilayer feedforward network: 10 convolutional layers with 512 3x3 filters
  - Feedforward NN with OptNet

$B \times 4 \times 4 \times 4 \longrightarrow B \times 64 \longrightarrow$ OptNet $\longrightarrow B \times 64 \longrightarrow B \times 4 \times 4 \times 4$

Input → Output

$B \times 4 \times 4 \times 4 \longrightarrow B \times 64 \longrightarrow$ Linear Mapping $\longrightarrow B \times n \longrightarrow$ OptNet $\longrightarrow B \times n \longrightarrow$ Linear Mapping $\longrightarrow B \times 64 \longrightarrow B \times 4 \times 4 \times 4$

Input → Output

OptNet

$$z^* = \min_z 0.5 \epsilon z^T z - z_i^T z$$

$$\text{s.t. } \boxed{Az = b}, z \geq 0$$

Application using OptNet

- Result
  - the convolutional is able to learn all of the necessary logic for the task and ends up over-fitting to the training data.
  - OptNet network learns most of the correct hard constraints within the first three epochs and is able to generalize much better to unseen examples.