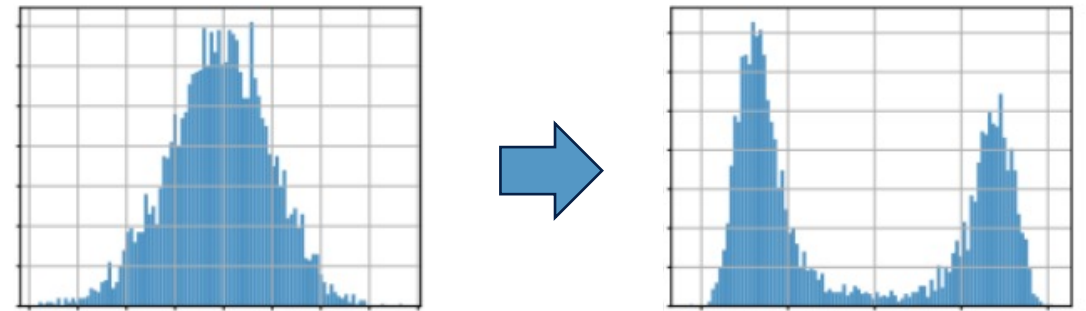


Cauchy-Schwarz Regularizers

Sueda Taner, Ziyi Wang, and Christoph Studer



Why are regularizer functions useful?

- Regularizer functions promote certain properties on solution $\hat{\mathbf{x}} \in \mathbb{R}^N$

$$\hat{\mathbf{x}} \in \operatorname{argmin}_{\mathbf{x} \in \mathbb{R}^N} f(\mathbf{x}) + \lambda \ell(\mathbf{x})$$

- $f(\mathbf{x})$: objective function
 - $\ell(\mathbf{x})$: regularizer function
 - λ : regularization parameter
-
- We propose *Cauchy-Schwarz (CS) regularizers*
 - simple
 - autoscaling
 - differentiable \rightarrow suitable for gradient-based numerical solvers
 - often invex (all stationary points are global minima)

Our recipe

- Recall CS inequality for $\mathbf{u}, \mathbf{v} \in \mathbb{R}^N$

$$|\langle \mathbf{u}, \mathbf{v} \rangle|^2 \leq \|\mathbf{u}\|^2 \|\mathbf{v}\|^2,$$

where equality holds if \mathbf{u} and \mathbf{v} are linearly dependent:

$$\mathbf{u} \sim \mathbf{v} \Leftrightarrow \exists (\beta_1, \beta_2) \in \mathbb{R}^2 \setminus \{(0,0)\} : \beta_1 \mathbf{u} = \beta_2 \mathbf{v}$$

- Inspired by the CS inequality, we propose

$$\ell(\mathbf{x}) \triangleq \|\mathbf{g}(\mathbf{x})\|^2 \|\mathbf{h}(\mathbf{x})\|^2 - |\langle \mathbf{g}(\mathbf{x}), \mathbf{h}(\mathbf{x}) \rangle|^2,$$

which is zero iff $\mathbf{x} \in \mathcal{X} \triangleq \{\tilde{\mathbf{x}} \in \mathbb{R}^N : \mathbf{g}(\tilde{\mathbf{x}}) \sim \mathbf{h}(\tilde{\mathbf{x}})\}$

Regularizer example #1: Symmetric binary (*bin*)

- We want $x_i \in \{-\alpha, \alpha\}, \forall i$ for $\alpha \in \mathbb{R}$; equivalently, for $\beta > 0$

$$\begin{bmatrix} x_1^2 \\ \vdots \\ x_N^2 \end{bmatrix} = \beta \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}$$

- Therefore, we set

$$\mathbf{g}(\mathbf{x}) \triangleq \mathbf{x}^{\circ 2}, \quad \mathbf{h}(\mathbf{x}) \triangleq \mathbf{1}_N$$

$$\Rightarrow \ell_{bin}(\mathbf{x}) \triangleq N \sum_n x_n^4 - \left(\sum_n x_n^2 \right)^2$$

Regularizer example #2: One-sided binary (*osb*)

- We want $x_i \in \{0, \alpha\}, \forall i$; equivalently,

$$\begin{bmatrix} x_1^2 \\ \vdots \\ x_N^2 \end{bmatrix} = \beta \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix}$$

- Therefore, we set

$$\mathbf{g}(\mathbf{x}) \triangleq \mathbf{x}^{\circ 2}, \quad \mathbf{h}(\mathbf{x}) \triangleq \mathbf{x}$$

$$\Rightarrow \ell_{osb}(\mathbf{x}) \triangleq \left(\sum_n x_n^2 \right) \left(\sum_n x_n^4 \right) - \left(\sum_n x_n^3 \right)^2$$

Regularizer example #3: Symmetric ternary (*ter*)

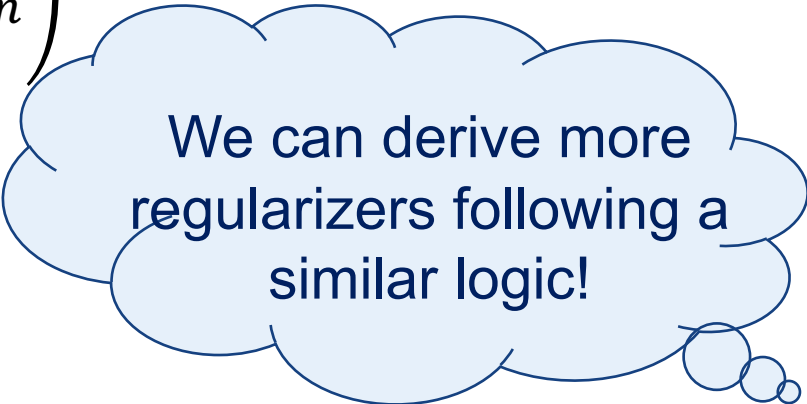
- We want $x_i \in \{-\alpha, 0, \alpha\}$ for $\alpha \in \mathbb{R}$; equivalently, for $\beta > 0$

$$\begin{bmatrix} x_1^3 \\ \vdots \\ x_N^3 \end{bmatrix} = \beta \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix}$$

- Therefore, we set

$$\mathbf{g}(\mathbf{x}) \triangleq \mathbf{x}^{\circ 3}, \quad \mathbf{h}(\mathbf{x}) \triangleq \mathbf{x}$$

$$\Rightarrow \ell_{ter}(\mathbf{x}) \triangleq \left(\sum_n x_n^2 \right) \left(\sum_n x_n^6 \right) - \left(\sum_n x_n^4 \right)^2$$



We can derive more regularizers following a similar logic!

Can we have regularizers beyond vector discretization?

Regularizer example #4: Eigenvector (*eig*)

- We set $\mathbf{g}(\mathbf{x}) \triangleq \mathbf{C}\mathbf{x}$, $\mathbf{h}(\mathbf{x}) \triangleq \mathbf{x}$, which gives

$$\ell_{eig}(\mathbf{x}) \triangleq \|\mathbf{C}\mathbf{x}\|^2 \|\mathbf{x}\|^2 - (\mathbf{x}^T \mathbf{C}\mathbf{x})^2$$

Regularizer example #5: Matrix with orthogonal columns (*om*)

- We set $\mathbf{g}(\mathbf{X}) \triangleq \text{vec}(\mathbf{X}^T \mathbf{X})$, $\mathbf{h}(\mathbf{X}) \triangleq \text{vec}(\mathbf{I}_K)$, which gives

$$\ell_{om}(\mathbf{X}) \triangleq K \|\mathbf{X}^T \mathbf{X}\|_F^2 - \|\mathbf{X}\|_F^4$$

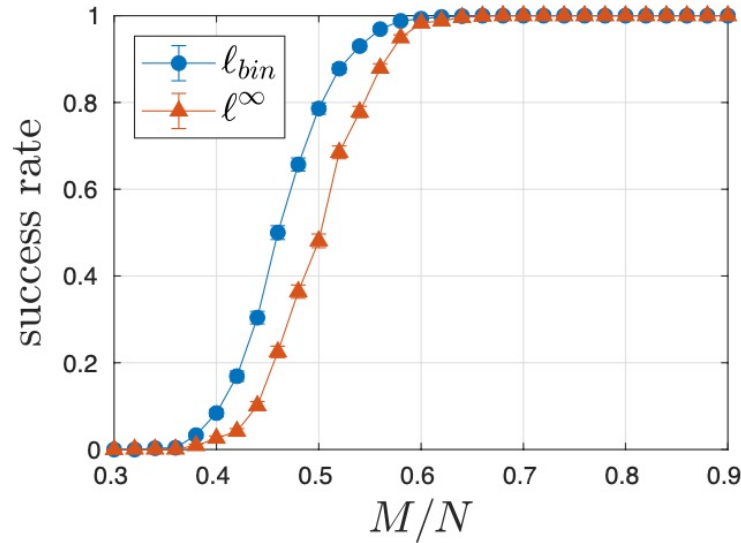
Application example #1: Recovering discrete-valued solutions

- Find discrete-valued solutions to underdetermined linear system of equations where $\mathbf{A} \in \mathbb{R}^{M \times N}$ is wide and full row rank

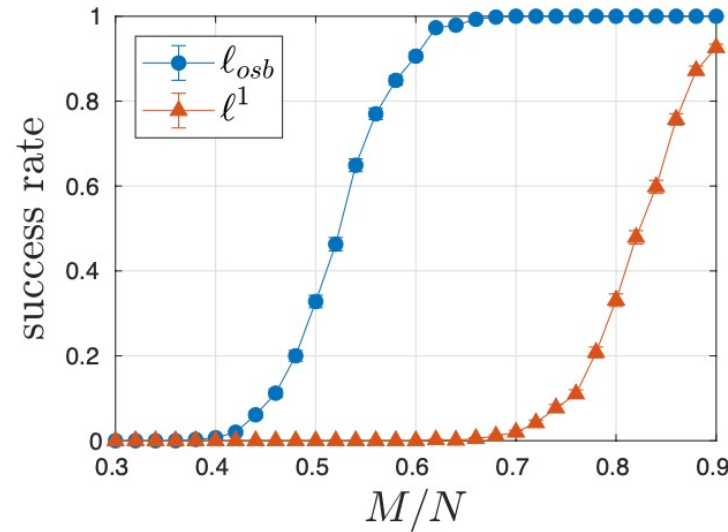
$$\hat{\mathbf{x}} \in \underset{\mathbf{x} \in \mathbb{R}^N}{\operatorname{argmin}} \ell(\mathbf{x}) \text{ subject to } \mathbf{Ax} = \mathbf{b} \quad (\text{P-rec})$$

- Constrained optimization problem
- To measure performance of using CS regularizers, we
 - draw normal random \mathbf{A} and random *bin/osb/ter* solution \mathbf{x}^* and calculate $\mathbf{b} = \mathbf{Ax}^*$
 - solve (P-rec) using projected gradient descent
 - measure success rate of recovering the true solution by averaging over trials

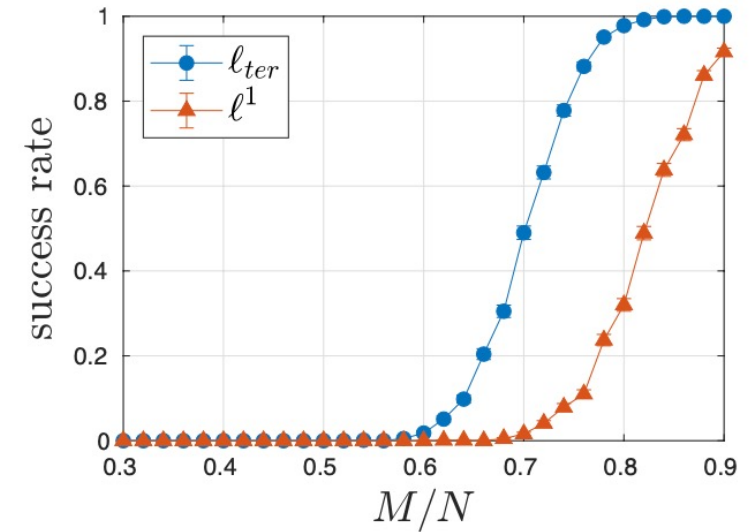
Application example #1: Results



(a) Symmetric binary



(b) One-sided binary

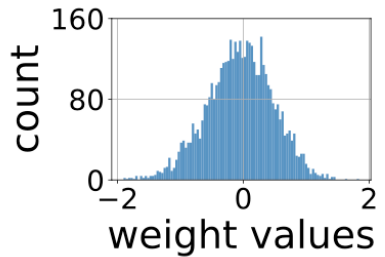


(c) Symmetric ternary

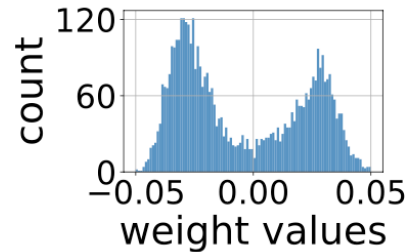
- CS regularizers have a higher success rate than ℓ^∞ -norm and ℓ^1 -norm baselines

Application example #2: Quantizing neural network weights

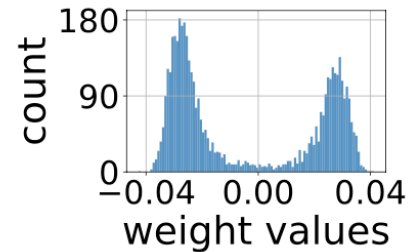
- Step 1: Regularized training of neural networks with ℓ_{bin} or ℓ_{ter}



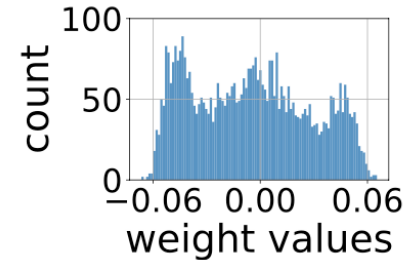
(a) Pretrained



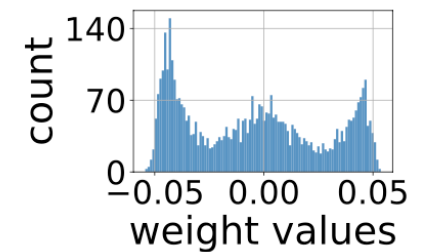
(b) ℓ_{bin} , 5 epochs



(c) ℓ_{bin} , 20 epochs



(d) ℓ_{ter} , 5 epochs



(e) ℓ_{ter} , 20 epochs

- Step 2: Quantizing the weights to binary or ternary scale factors
- Step 3: Training with quantized weights (fix the signs of weights; continue training shared scale factors, biases, BN parameters, etc.)

Application example #2: Results for image classification on ImageNet

- Comparable accuracy to SOTA baselines
- CS regularizers have storage and complexity advantage during training

Binarized ResNet-18

Method	Top-1 (%)
ResNet-18 (FP)	69.8
SQ-BWN (Dong et al., 2017)	58.4
BWN (Rastegari et al., 2016)	60.8
HWGQ (Cai et al., 2017)	61.3
PCCN (Gu et al., 2019)	63.5
BWHN (Hu et al., 2018)	64.3
ADMM (Leng et al., 2018)	64.8
IR-Net (Qin et al., 2020)	66.5
LCR-BNN (Shang et al., 2022)	66.9
DAQ (Kim et al., 2021)	67.2
ProxyBNN (He et al., 2020)	67.7
Ours (ℓ_{bin})	62.8 \pm 0.09

Ternarized ResNet-18

Method	Top-1 (%)
ResNet-18 (FP)	69.8
TWN (Li et al., 2016)	61.8
SQ-TWN (Dong et al., 2017)	63.8
QNet (Yang et al., 2019)	66.5
ADMM (Leng et al., 2018)	67.0
LQ (Zhang et al., 2018)	68.0
QIL (Jung et al., 2019)	68.1
Ours (ℓ_{ter})	65.3 \pm 0.08