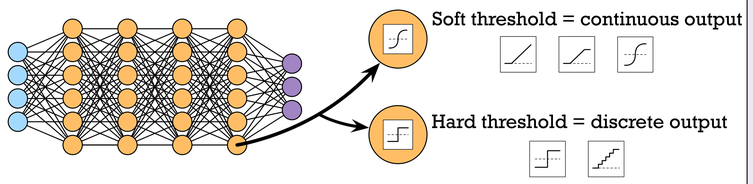
Deep Learning as a Mixed Convex-Combinatorial Optimization Problem

Abram L. Friesen and Pedro Domingos
{afriesen, pedrod}@cs.washington.edu

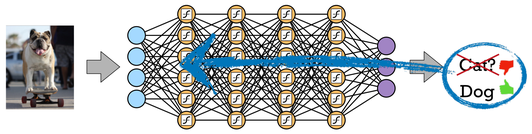
Hard-Threshold Activations Enable Scaling



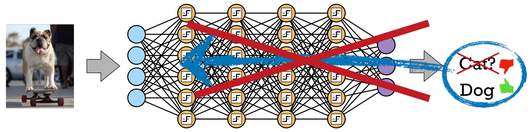
Low energy and memory requirements.
Fast computation.
Alleviate vanishing & exploding gradients.
Resistant to adversarial attacks; e.g., Galloway et al. (ICLR 2018).
Crucial for developing large systems of deep networks.

No Backprop in Hard-Threshold Networks

Soft-threshold network:

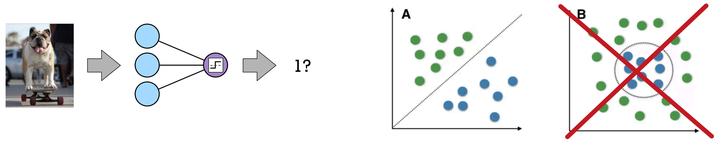


Hard-threshold network:



Derivative of hard-threshold activations is zero almost everywhere.

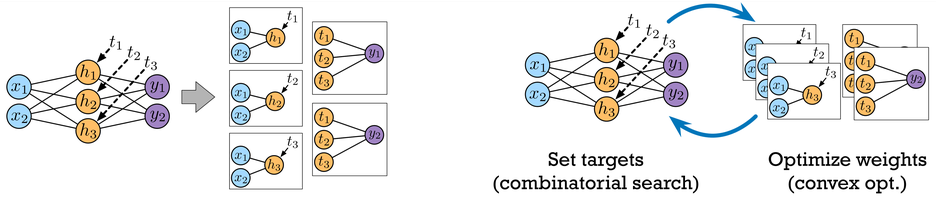
Perceptrons are Hard-Threshold Networks



Learnable without backpropagation.
Not expressive; can only learn linearly-separable datasets.

Learning Multi-Layer Models with Hard-Thresholds

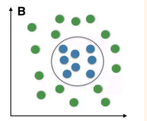
Idea: If we knew the value each h_i *should* take, then network would decompose into individual perceptrons.



How to set targets? Define and use *feasibility*.

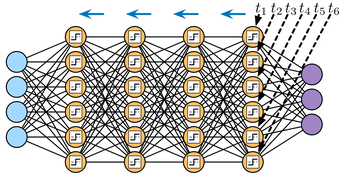
Targets T are feasible *for a dataset D iff all perceptrons in the network are linearly-separable given T.*

Learned network does *not* require linearly-separable dataset.



Proposition.
Let D be a dataset and T be feasible targets for network $f(X; W)$ where each layer's weights W_d were trained independently with inputs T_{d-1} and targets T_d. Then f will correctly classify each instance in D.

Feasible Target Propagation



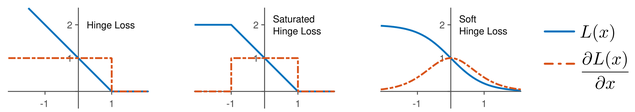
Algorithm 2 Train an ℓ-layer hard-threshold network $Y = f(X; W)$ on dataset $\mathcal{D} = (X, T_\ell)$ with mini-batch feasible target propagation (FTPROP-MB) using loss functions $L = \{L_d\}_{d=1}^{\ell}$.

initialize weights $W = \{W_1, \ldots, W_\ell\}$ randomly
for each minibatch (X_b, T_b) from $\mathcal{D}$ **do**
  initialize targets $T_1, \ldots, T_{\ell-1}$ as the outputs of their hidden units in $f(X_b; W)$ *// forward pass*
  set $T_0 \leftarrow X_b$, set $T_\ell \leftarrow T_b$, and set $T \leftarrow \{T_0, \ldots, T_\ell\}$
  FTPROP-MB(W, T, L, ℓ)

function FTPROP-MB(weights W, targets T, losses L, and layer index d)
  $\hat{T}_{d-1} \leftarrow$ set targets for upstream layer based on current weights W_d and loss $L_d(Z_d, T_d)$
  update W_d with respect to layer loss $L_d(Z_d, T_d)$ *// where $Z_d = W_d T_{d-1} = W_d H_{d-1}$*
  if $d > 1$ **then** FTPROP-MB$(W, \{T_0, \ldots, \hat{T}_{d-1}, \ldots, T_\ell\}, L, d-1)$

Each iteration, set targets recursively and then independently update each layer's weights.
Combinatorial search is hard, so define per-layer loss functions and set targets heuristically.

Layer Loss Functions



Prioritize certain targets: Scale each target by the magnitude of its partial derivative w.r.t. the next layer's loss.

$$L_d(z_{dj}, t_{dj}) = \text{hinge}(z_{dj}, t_{dj}) \cdot \left| \frac{\partial L_{d+1}}{\partial h_{dj}} \right|$$

Target Heuristic

$$\underbrace{t(h_{dj})}_{\text{target}} \triangleq \text{sign}\left(-\underbrace{\frac{\partial}{\partial h_{dj}}}_{\text{derivative}} \underbrace{L_{d+1}(H_{d+1}, T_{d+1})}_{\text{next layer loss}} \right)$$

Relationship to Straight-Through Estimator

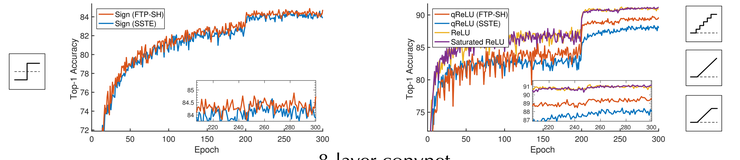
No existing justification for the straight-through estimator (STE) [Hinton (2012)].
We show that different STEs correspond to different per-layer loss choices in FTPROP.

Original STE: [Hinton (2012)] $L(t, z) = -tz$

Saturated STE: [Hubara et al. (2016)] $L(t, z) = \max(0, 1 - \max(tz, -1))$ = Saturated Hinge

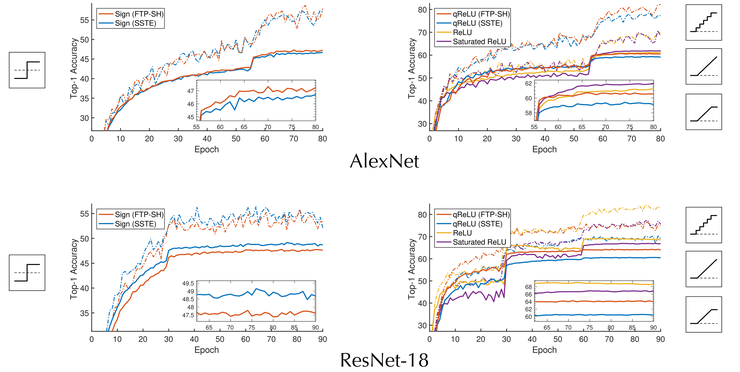
Experiments: Image Classification

CIFAR-10



8-layer convnet

ImageNet

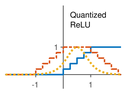


AlexNet

ResNet-18

All Results

	Sign		qReLU		Baselines	
	SSTE	*FTP-SH*	*SSTE*	*FTP-SH*	*ReLU*	*Sat. ReLU*
4-layer convnet (CIFAR-10)	80.6	**81.3**	85.6	85.5	86.5	87.3
8-layer convnet (CIFAR-10)	84.6	**84.9**	88.4	**89.8**	91.2	91.2
AlexNet (ImageNet)	46.7	**47.3**	59.4	**60.7**	61.3	61.9
ResNet-18 (ImageNet)	**49.1**	47.8	60.6	**64.3**	69.1	66.9



Paul G. Allen School of Computer Science and Engineering
University of Washington, Seattle WA, USA

Code is available at github.com/afriesen/ftprop