Input-Convex Deep Networks

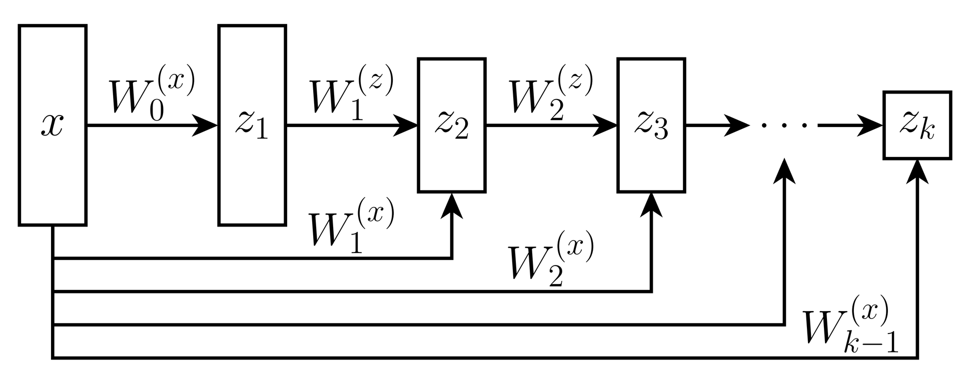
Brandon Amos and J. Zico Kolter School of Computer Science, Carnegie Mellon University

Introduction

- We introduce a new neural network architecture:
  - **Input-Convex Neural Networks (ICNNs)**
- **Definition:** Scalar-valued neural network $f(x;\theta)$
  - f is **convex** in the input x
  - (f is not convex in the parameters $\theta = \{W_i, b_i\}$)
- Model allows **global** optimization over some of the inputs to the network, given fixed values for other inputs
- Many existing neural-network architectures can be "easily" made input-convex

Input-Convex Neural Networks

Typical ICNN model:



$$z_{i+1} = g_i\left(W_i^{(z)} z_i + W_i^{(x)} x + b_i\right) \quad i = 0,\dots,k-1$$

$$f(x;\theta) = z_k$$

- z_i are the **layer activations** (with $z_0 \equiv 0$)
- g_i are non-linear **activation functions**
- Also supports linear operations like convolutions

Proposition 1. *The function f is convex in x provided that all $W_{1:k-1}^{(z)}$ are non-negative, and all functions g_i are convex and non-decreasing*

- Many common non-linearities g_i (e.g., (PL)ReLU and max-pooling) are already convex and non-decreasing
- Non-negativity of $W^{(z)}$ terms is a notable restriction
- Joint convexity in all inputs also restrictive (can be extended to partial convexity, which then generalizes ICNNs and traditional feedforward networks)

ICNN Use Cases

- Structured prediction
  - Similar model to Belanger and McCallum [1] (non-convex deep networks for structured prediction)
  - Network takes input and output pairs: $f(x,y;\theta)$
  - Inference for an input x:

$$\hat{y} = \underset{y}{\operatorname{argmin}}\, f(x,y;\theta)$$

  (for ICNNs, a convex, thus globally solvable problem)
- Exemplars in learning
  - Same setting as above, but also inference over x

$$f(x_k^*, y = e_k; \theta) \le \min_x f(x, y = e_k; \theta)$$

- Data imputation*
  - Infer missing values from values that are present
  - $\hat{x}_{\mathcal{I}} = \operatorname{argmin}_{x_{\mathcal{I}}} f(x_{\mathcal{I}}, x_{\neg\mathcal{I}}; \theta)$
- Reinforcement learning*
  - Represent $Q(s,a;\theta)$ function as a (negated) ICNN
  - Finding best action $\operatorname{argmax}_a Q(s,a;\theta)$ (even for continuous action spaces) is a convex problem

*Work in progress

ICNN Inference

- In general, inference requires optimization over some inputs given other inputs (always a convex problem!)
  - E.g. structured prediction: $\hat{y} = \operatorname{argmin}_y f(x,y;\theta)$
- For ICNNs with ReLUs, max pooling, fully connected units, and convolutions, inference is a **linear program**

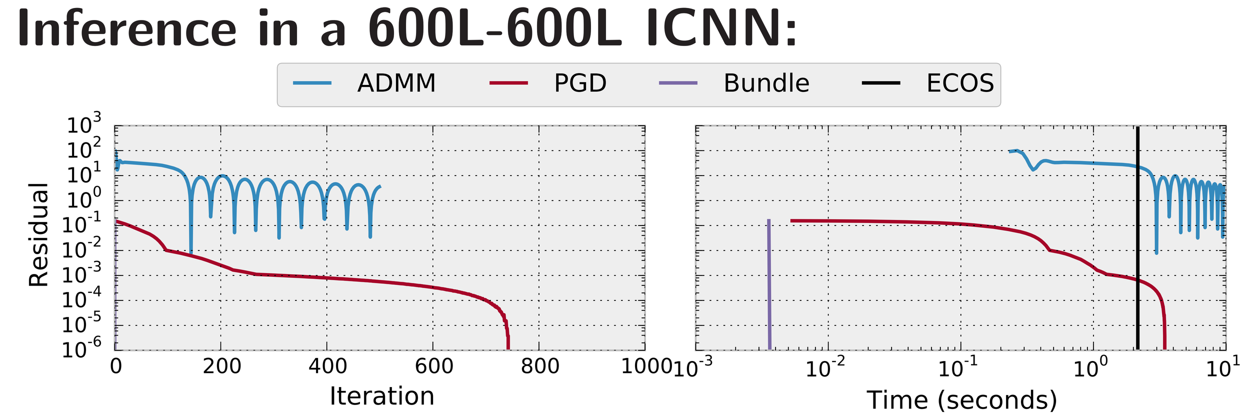
$$\min_{y,z_1,\dots,z_k} \; z_k \quad \text{s.t.} \quad z_{i+1} \ge W_i^{(z)} z_i + W_i^{(xy)} \begin{bmatrix} x \\ y \end{bmatrix} + b_i, \quad \forall i$$

$$z_i \ge 0, \quad \forall i \ne k$$

Solution approaches:

- Full LP formulation (variable for each hidden unit)
  - ADMM or an off-the-shelf solver (like ECOS)
- Gradient-based methods
  - Gradient descent, bundle and cutting plane methods

Inference in a 600L-600L ICNN:



ICNN Learning

- Can train networks using framework similar to max-margin structured prediction [4, 3]
- E.g., in structured prediction setting, want to find θ such that for all training inputs (x_i, y_i)

$$f(x_i, y_i; \theta) \le \min_{y\in\mathcal{Y}} \left(f(x_i, y; \theta) - \Delta(y_i, y)\right)$$

- $\Delta(y_i, y)$ is a *margin-scaling* term
  - Margin for the inequality when y_i different from y
  - In multi-class classification: $\mathcal{Y}$ is simplex and $\Delta(y_i, y) = y^T(1 - y_i)$
- Note: training network is **not** a convex problem

Subgradient method for structured prediction [2]:

- Training example x_i, y_i
- Solve $y^\star = \operatorname{argmin}_{y\in\mathcal{Y}} f(x_i, y; \theta) - \Delta(y_i, y)$
- If margin is violated, update

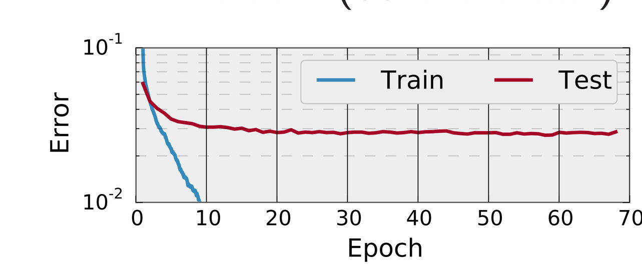
$$\theta \leftarrow \mathcal{P}_+\left[\theta - \alpha\left(\lambda\theta + \nabla_\theta f(x_i, y_i, \theta) - \nabla_\theta f(x_i, y^\star; \theta)\right)\right]$$

where $\mathcal{P}_+$ projects $W_{1:k-1}^{(z)}$ onto the non-negative orthant

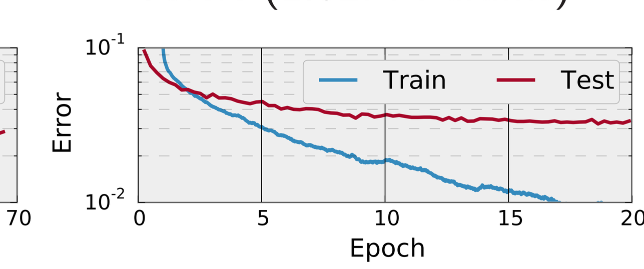
Experiment: MNIST Classification

Fully Connected (600L-600L)

Feedforward (837k Params)

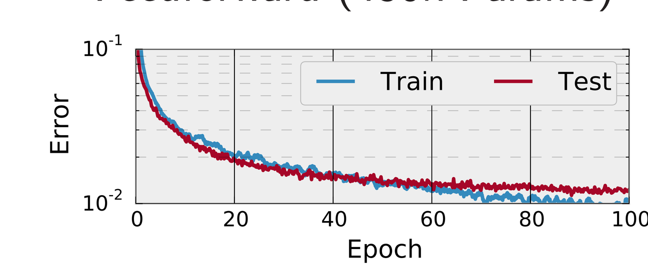


ICNN (1.32M Params)

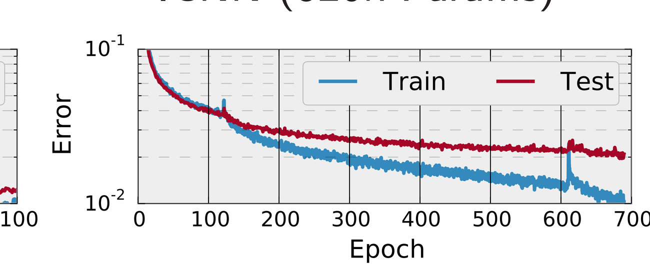


LeNet (20C-MP-50C-MP-500L, 5×5 conv)

Feedforward (430k Params)



ICNN (826k Params)



Error summary at last iteration:

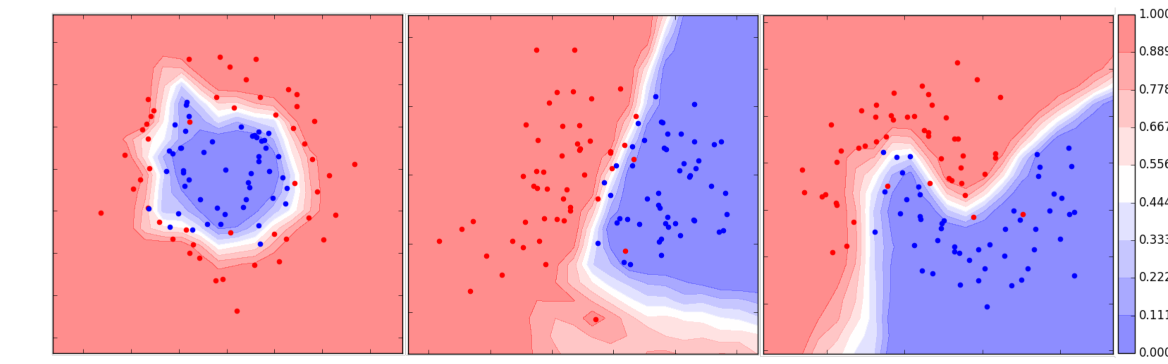
		Train	Test
Feedforward	Fully Connected	$3.3 \cdot 10^{-5}$	0.029
	LeNet	0.010	0.012
ICNN	Fully Connected	0.0075	0.034
	LeNet	0.010	0.021

CUDA LeNet runtimes:

- **Training minibatch** with 128 instances

Feedforward	0.038 ± 0.006 seconds
ICNN	0.302 ± 0.010 seconds

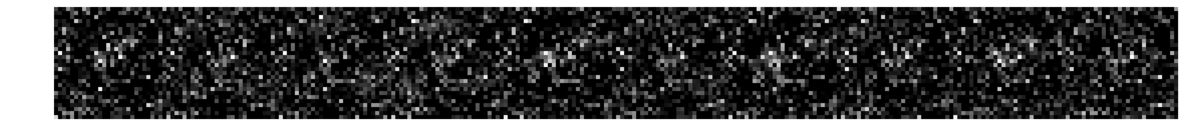
Experiment: Synthetic Classification



- 2-layer linear ICNN with ReLU (200 units per layer)
- ICNNs can learn non-convex decision boundaries

Experiment: Exemplar Learning

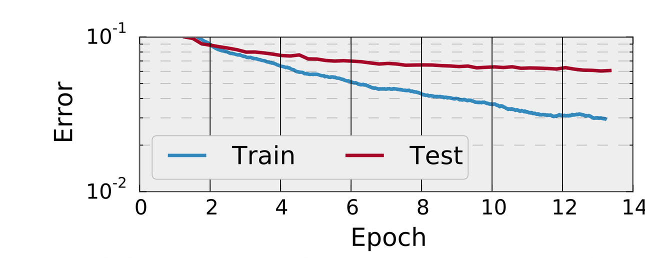
- Consider each class in a fully connected MNIST ICNN
  - Network from MNIST results with 1.32M params
  - $\min_x f(x, y = k; \theta)$ of the trained network on digits:

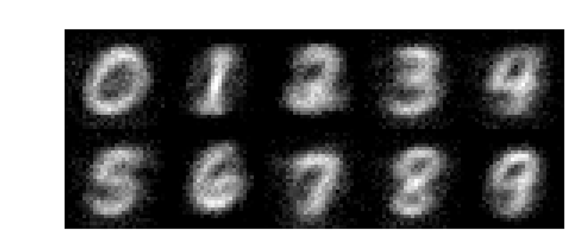


- **Regularization idea:** Jointly optimize y and x
  - In classification, average the examples for each class
  - Represent the exemplar for class i as x_*^i
- Use margin scaling term $\Delta(x_*^i, x) = \frac{\gamma}{2}||x - x_*^i||_2^2$
  - Requires that we use $\tilde{f} \equiv f(x,y) + \frac{\gamma}{2}||x||_2^2$ to maintain convexity in the augmented inference problem

MNIST classification with exemplar learning:

- In each minibatch, learn all 10 exemplars and 128 classification samples





- Network learns exemplars at the expense of accuracy

References

[1] D. Belanger and A. McCallum. "Structured Prediction Energy Networks". In: *arXiv:1511.06350* (2015).

[2] N. Ratliff, J. Bagnell, and M. Zinkevich. "(Approximate) Subgradient Methods for Structured Prediction". In: *ICAIS*. 2007.

[3] B. Taskar et al. "Learning structured prediction models: A large margin approach". In: *ICML*. 2005.

[4] I. Tsochantaridis et al. "Large margin methods for structured and interdependent output variables". In: *JMLR* (2005).