Input Convex Neural Networks

Brandon Amos Lei Xu* J. Zico Kolter
Carnegie Mellon University
School of Computer Science
*Tsinghua University, work done while at CMU

ICML 2017

Big picture

What are the “atomic operations” or building blocks of modern AI systems?

The current situation: Matrix-vector products (dense or sparse/structured), (sub)differentiable non-linear functions, random sampling

This talk: We should consider (convex) optimization as another potential atomic operation, to be composed with others

Note: we already use optimization in the learning procedures, but we should also consider it as an operation for inference and control

This is not a new idea: some of the original work in neural networks considered them composition with more complex, optimization-based prediction (c.f. structured prediction)

Applications of Optimization for Inference

Structured prediction: define a network over $\mathcal{X} \times \mathcal{Y}$ and predict via

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

Data imputation: build a network over only over $\mathcal{Y}$, given $y_{\mathcal{I}}$ populate the remaining entries via

$$\hat{y}_{\bar{\mathcal{I}}} = \operatorname{argmin}_{y_{\bar{\mathcal{I}}}} f(y_{\bar{\mathcal{I}}}, y_{\mathcal{I}}; \theta)$$

Continuous action reinforcement learning: Represent Q function as $Q^\star(s, a) = -f(s, a; \theta)$, policy becomes

$$\pi^\star(s) = \operatorname{argmin}_a f(s, a; \theta)$$

Talk Overview

1. Our Contribution: Input Convex Neural Networks
2. Challenges: Inference and Learning
3. Experiments
    1. Synthetic
    2. Multi-label Classification
    3. Image Completion
    4. Continuous–Action Q-Learning

Input Convex Neural Networks (ICNNS)

Scalar-valued network $f(x, y; \theta)$ such that f is convex in y for all values of x (note that these networks are still not convex in $\theta = \{W_i, b_i\}$)

ICNNs can efficiently optimize over some inputs to the network given other inputs

Efficiently captures dependencies in the output space for prediction

It turns out, we don't need very many restrictions on the network to achieve this property

Applications of Optimization for Inference

With ICNNs: All of these problems are convex, "easy" to solve globally

Structured prediction: define a network over $\mathcal{X} \times \mathcal{Y}$ and predict via

$$\hat{y}(x) = \text{argmin}_y \, f(x, y; \theta)$$

Data imputation: build a network over only over $\mathcal{Y}$, given $y_{\mathcal{I}}$ populate the remaining entries via

$$\hat{y}_{\bar{\mathcal{I}}} = \text{argmin}_{y_{\bar{\mathcal{I}}}} \, f(y_{\bar{\mathcal{I}}}, y_{\mathcal{I}}; \theta)$$

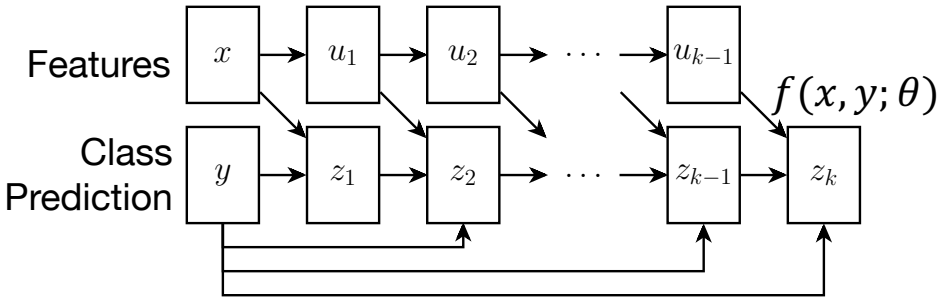
Continuous action reinforcement learning: Represent Q function using ICNN $Q^\star(s, a) = -f(s, a; \theta)$, policy becomes

$$\pi^\star(s) = \text{argmin}_a f(s, a; \theta)$$

Example Networks

ICNN for structured prediction:

$$\hat{y}(x) = \text{argmin}_y \, f(x, y; \theta)$$



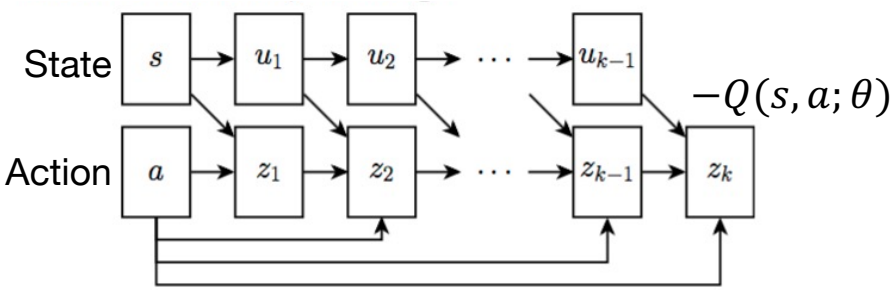
$$u_{i+1} = \tilde{g}_i(\tilde{W}_i u_i + \tilde{b}_i)$$

$$z_{i+1} = g_i\left(W_i^{(uz)}(u_i \circ z_i) + W_i^{(u)} u_i + W_i^{(z)} z_i + W_i^{(y)} y_i + b_i\right)$$

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

ICNN for Q learning:

$$\pi^\star(s) = \text{argmin}_a \, -Q(s, a; \theta)$$



$$u_{i+1} = \tilde{g}_i(\tilde{W}_i u_i + \tilde{b}_i)$$

$$z_{i+1} = g_i\Big(W_i^{(z)}\big(z_i \circ [W_i^{(zu)} u_i + b_i^{(z)}]_+\big) + W_i^{(a)}\big(a \circ (W_i^{(au)} u_i + b_i^{(a)})\big) + W_i^{(u)} u_i + b_i\Big)$$

$$-Q(s, a; \theta) = f(s, a; \theta) = z_k, \; u_0 = s, \; z_0 = a$$

How to achieve input convexity?

Most networks can be "trivially" modified to guarantee input convexity

Consider simple feedforward ReLU network:

$$z_{i+1} = \max\{0, W_i z_i + b_i\}, \qquad i = 1, \ldots, k$$
$$f(y; \theta) = z_{k+1},\ z_1 = y$$

f is convex in y provided that the W_i are non-negative for $i > 1$

More generally, any activation function that is convex and non-decreasing also has this property.

Is convexity restrictive?

Yes (by definition, the functions are restricted to be convex), but not that bad in practice

E.g., we can trivially capture any feedforward network $\tilde{f}(x)$ with the network $f(x, y) = \left(y - \tilde{f}(x)\right)^2$

More complex convex portion adds additional structure over y, which can still be "easily" optimized over

We'll see more evidence for this later

Talk Overview

1. Our Contribution: Input Convex Neural Networks
2. Challenges: Inference and Learning
3. Experiments
    1. Synthetic
    2. Multi-label Classification
    3. Image Completion
    4. Continuous–Action Q-Learning

Challenges for ICNNs

Inference: how do we efficiently perform the optimization?

$$y^\star(x;\theta) = \mathrm{argmin}_y \, f(x, y; \theta)$$

Learning: How do we train the network (find θ) such that it gives good predictions?

$$\mathrm{minimize}_\theta \sum_{i=1}^{n} \ell\big(y_i, y^\star(x_i;\theta)\big)$$

Inference in ICNNs

In theory, inference in ICNNs is just a linear program

$$\begin{aligned} \min_y f(y;\theta) = \min_{y,z}\ & z_{k+1} \\ \text{s.t. } & z_{i+1} \geq W_i z_i + b_i \\ & z_i \geq 0 \ \text{ for } i > 1 \\ & z_1 = y \end{aligned}$$

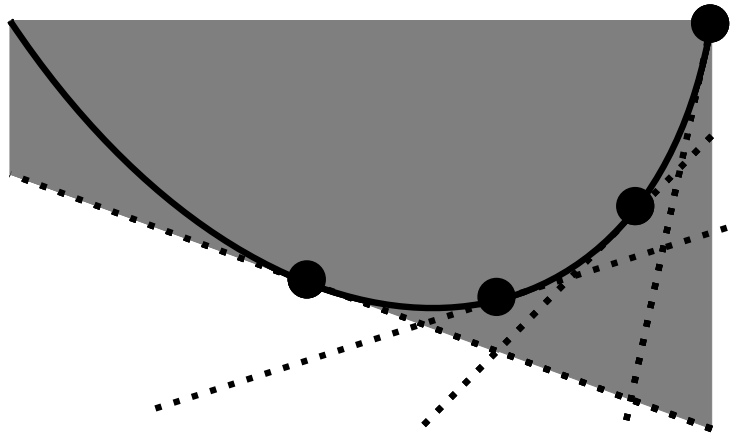
This program has as many variables as hidden units in the network, exact solution methods require that we invert the $W_i^T W_i$ matrices

Instead, exploit the fact that we can easily compute the gradient of $f(x, y; \theta)$ with respect to y (this is just backprop), optimize using gradient-based methods.

We found that the bundle method performs better than gradient descent in some cases.

Inference with the Bundle Method

Repeatedly minimize a lower bound on the function



Using convexity property to minimize more quickly than gradient descent

Boundary constraints are difficult, so we actually use an entropy penalty

$$\tilde{f}(x, y; \theta) + y \log y + (1 - y) \log(1 - y)$$

ICNN Learning

Two possibilities for training networks

1. **Max-margin structured prediction**: enforce constraint that

$$f(x_i, y_i; \theta) \leq \operatorname{argmin}_y \left(f(x_i, y; \theta) + \Delta(y, y_i) \right)$$

   Common structured prediction approach

   Margin-scaling term $\Delta(y, y_i)$ can be finicky

2. Direct **argmin differentiation**, directly compute

$$\nabla_\theta \ell\left(y_i, y^\star(x_i; \theta)\right)$$

   Can be approximated by unrolling an optimization procedure

   Plays nicely with bundle method and approximate optimization

   May require some differential calculus (nothing too nasty)

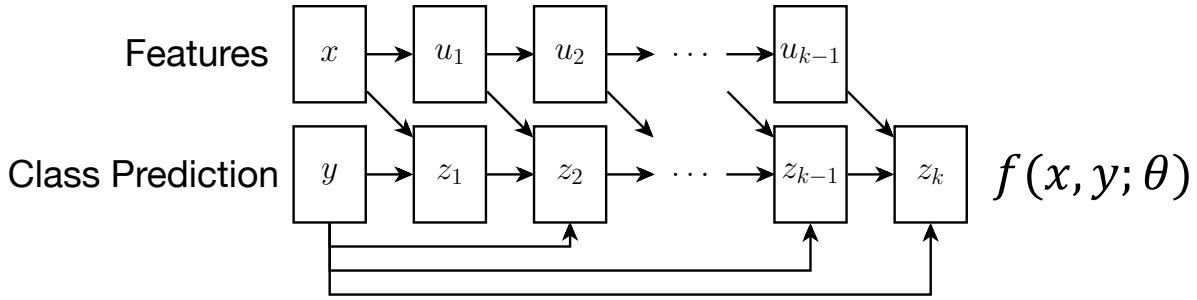
Talk Overview

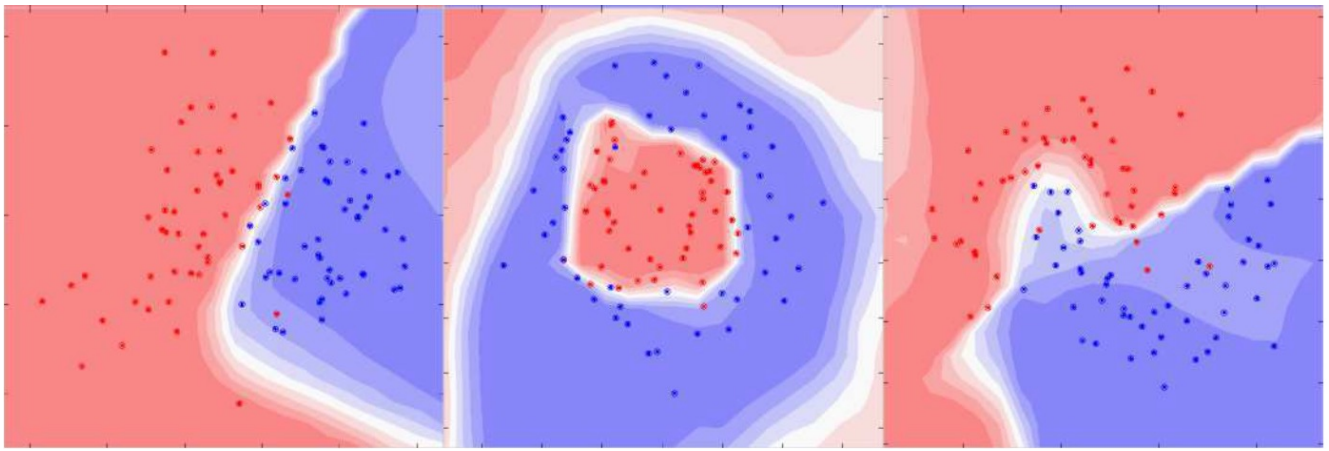
1. Our Contribution: Input Convex Neural Networks
2. Challenges: Inference and Learning
3. Experiments
    1. Synthetic
    2. Multi-label Classification
    3. Image Completion
    4. Continuous–Action Q-Learning

Results: toy example

Partially input convex neural network trained to classify points in 2D space

$$\hat{y}(x) = \text{argmin}_y \, f(x, y; \theta)$$





Only point to remember from this: convex energy function does *not* imply a convex decision boundary; argmin operator is a powerful one

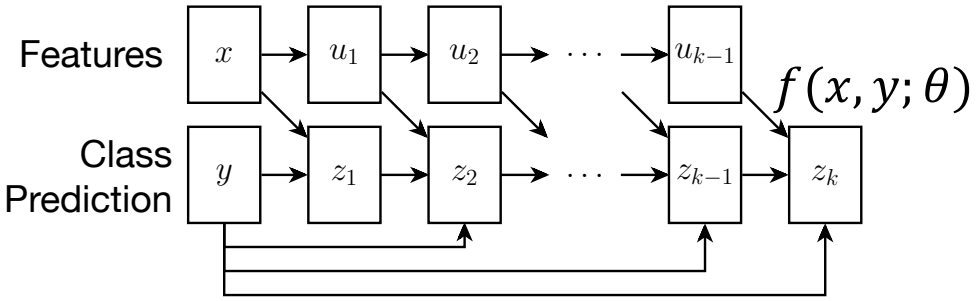
Results: multi-label classification

Task: Predict tags for bibtex entries from bag of words features

Used in Belanger and McCallum, 2016: Structured Prediction Energy Networks

ICNNs almost recover the same performance as SPENs despite the convexity restrictions

$$\hat{y}(x) = \text{argmin}_y \, f(x, y; \theta)$$



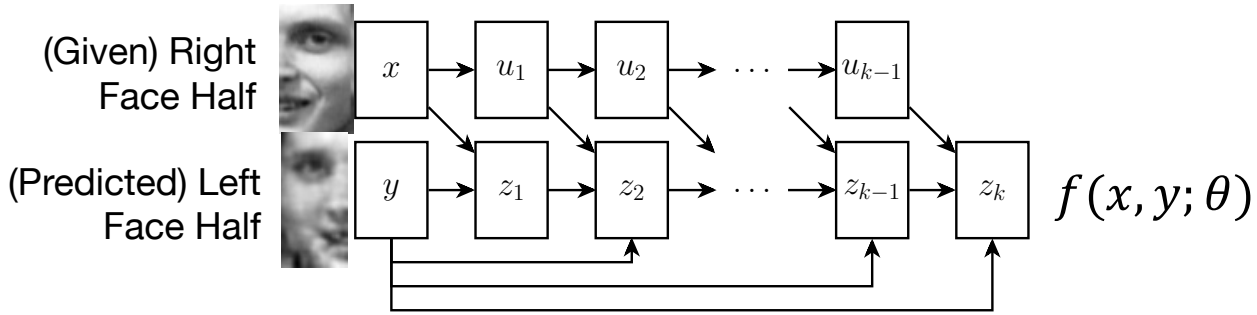
Method	Test Macro-F1
NN (Baseline)	0.396
SPEN	0.422
ICNN	0.415

Results: image completion

Task: Predict the left side of the image given the right side. Used in Poon and Domingos 2011; Sum-Product Networks

ICNN: DQN-like network over both input and output

$$\hat{y}(x) = \text{argmin}_y \, f(x, y; \theta)$$



Method	MSE
SPN	942
ICNN: Bundle Entropy	833.0
ICNN: Gradient Descent	872.0
ICNN: Non-convex (GD)	850.9

ICNN Test Set Completions

Results: continuous-action RL

Using Q-learning with entropy-regularized ICNN to represent Q function.

Comparing to DDPG [Lillicrap et al., 2016], and NAF [Gu et al., 2016]

NAF is particularly similar: also uses a value function that is convex in the input, but uses a quadratic representation to guarantee this convexity

Select actions with:

$$a^\star(s) = \operatorname{argmin}_a -Q(s,a;\theta)$$

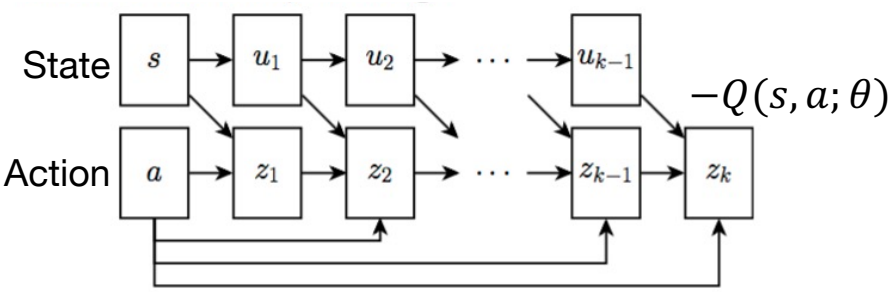


Table: Max (rolling) test reward obtained on Gym continuous control tasks

Task	DDPG	NAF	ICNN
Ant	1000.00	999.03	**1056.29**
HalfCheetah	2909.77	2575.16	**3822.99**
Hopper	**1501.33**	1100.43	831.00
Humanoid	524.09	**5000.68**	433.38
HumanoidStandup	134265.96	116399.05	**141217.38**
InvDoubPend	**9358.81**	**9359.59**	**9359.41**
InvPend	**1000.00**	**1000.00**	**1000.00**
Reacher	-6.10	-6.31	**-5.08**
Swimmer	49.79	**69.71**	64.89
Walker2d	**1604.18**	1007.25	298.21

Input Convex Neural Networks

Brandon Amos Lei Xu J. Zico Kolter
Carnegie Mellon University
School of Computer Science

1. Our Contribution: Input Convex Neural Networks
2. Challenges: Inference and Learning
3. Experiments
    1. Synthetic
    2. Multi-label Classification
    3. Image Completion
    4. Continuous–Action Q-Learning

The full TensorFlow source code to reproduce all of our experiments is available online at https://github.com/locuslab/icnn