INPUT-CONVEX DEEP NETWORKS

Brandon Amos, J. Zico Kolter
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
{bamos,zkolter}@cs.cmu.edu

ABSTRACT

This paper introduces a new class of neural networks that we refer to as *input-convex* neural networks, networks that are convex in their inputs (as opposed to their parameters). We discuss the nature and representational power of these networks, illustrate how the prediction (inference) problem can be solved via convex optimization, and discuss their application to structured prediction problems. We highlight a few simple examples of these networks applied to classification tasks, where we illustrate that the networks perform substantially better than any other approximator we are aware of that is convex in its inputs.

1 INTRODUCTION

The majority of current deep network architectures for supervised learning problems employ a feed-forward inference step: given some input x the prediction $\hat{y}$ is computed by evaluating

$$\hat{y} = f(x;\theta) \tag{1}$$

where f is some neural network function and θ denotes parameters governing the function. This paper focuses instead on the structured prediction setting, also referred to as energy-based learning (LeCun et al., 2006), where instead we define a joint function over inputs and outputs $f(x, y; \theta)$, and predictions are made via an optimization procedure

$$\hat{y} = \operatorname*{argmin}_{y \in \mathcal{Y}} f(x, y; \theta). \tag{2}$$

These subsume feedforward architectures (we could simply define $f(x, y; \theta) = \|y - f'(x; \theta)\|$ and the minimization would return $f'(x; \theta)$), but additionally allows for richer joint representations over x and y. The downside is that computing the predictions now requires optimizing over the $f(x, y; \theta)$ function itself, which can be challenging if f is a function of x and y jointly.

This paper proposes a new approach to structured prediction in deep networks based upon what we refer to as *input-convex* neural networks (ICNNs).[1] Simply put, these are neural network architectures of the type (2) where $f(x, y; \theta)$ is convex in y (or perhaps even jointly convex in x and y). As such, they encompass a class of function approximators where it *is* easy (in some sense of the word), to globally compute the minimization in (2). A primary claim of the current work is a practical one, that we can build input-convex neural network architectures very similar in form to existing state-of-the-art network architectures. We describe how their use in structured prediction, and highlight some results on a few simple classification tasks, where they thus far obtain modest accuracy, which is nonetheless higher than existing classifiers we are aware of that are convex in their inputs.

2 INPUT-CONVEX NEURAL NETWORKS

As with most neural networks, there are several possible architectures to consider but for simplicity we focus here on a single case that provides convexity in both x and y terms (a substantially

[1]We use this term instead of "convex neural network" because convexity in machine learning typically refers to convexity in the *parameters*. In contrast, the networks we discuss are definitely not convex in the parameters θ, and thus training the systems remains a non-convex problem.

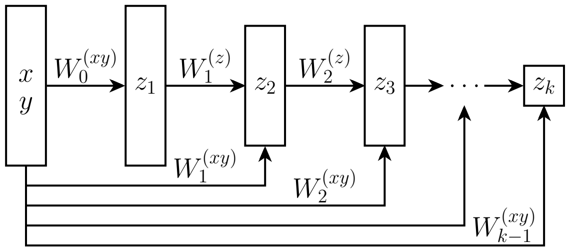


Figure 1: An example input-convex neural network architecture (here, convex in both x and y).

more restrictive class than networks that are only convex in y). We define a k-layer neural network architecture over input/output pairs (x, y) by the recurrence

$$\begin{aligned} z_{i+1} &= g_i\left(W_i^{(z)} z_i + W_i^{(xy)} \begin{bmatrix} x \\ y \end{bmatrix} + b_i\right), \;\; i = 0, \ldots, k-1 \\ f(x, y; \theta) &= z_k \end{aligned} \tag{3}$$

where z_i denote the layer activations (with $z_0, W_0^{(z)} \equiv 0$), $\theta = \{W_{0:k-1}^{(xy)}, W_{1:k-1}^{(z)}, b_{0:k-1}\}$ are the parameters, and g_i are non-linear activation functions. Other linear operators like convolutions could be included, without changing the convexity properties. We then have the following property:

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

The proof is simple and follows from the fact that non-negative sums of convex functions are also convex and that the composition of a convex and convex non-decreasing function is also convex (see e.g. Boyd & Vandenberghe (2004, 3.2.4)). The constraint that the g_i be convex non-decreasing is not particularly restrictive, as current non-linear activation units like the rectified linear unit or max-pooling unit already satisfy this constraint. The constraint that the $W^{(z)}$ terms be non-negative is indeed a substantial restriction over traditional neural networks, precisely because they enforce the ICNNs to be convex with respect to their inputs, whereas general neural networks are general function approximators.

Prediction Once a ICNN has been fit to data (as discussed below), we solve the prediction (inference) problem by directly solving the optimization problem (2). For example, in the case of an ICNN where g_i are rectified linear units $g_i(z) = \max\{0, z\}$ for layers $1, \ldots, k-1$ and a linear unit in layer k, this takes the form of the linear program

$$\begin{aligned} \underset{y, z_1, \ldots, z_k}{\text{minimize}} \;\; & z_k \\ \text{subject to} \;\; & z_{i+1} \geq W_i^{(z)} z_i + W_i^{(xy)} \begin{bmatrix} x \\ y \end{bmatrix} + b_i, \;\; i = 0, \ldots, k-1 \\ & z_i \geq 0, \;\; i = 1, \ldots, k-1 \end{aligned} \tag{4}$$

Although larger problems will likely require specialized solvers to solve these optimization problems, in this current work we focus on examples small enough where these optimization problems can be solved with off-the-shelf linear programming methods.

Learning The ICNN is an instance of a problem where "MAP inference" (i.e., solving the optimization problem (2)) is easily accomplished, but probabilistic inference (i.e., computing the partition function for a distribution defined by $p(y|x) = \exp(-f(x, y)))$ is non-trivial. This motivates training such models using a max-margin structured prediction framework (Tsochantaridis et al., 2005; Taskar et al., 2005). In particular, given some training set (x_i, y_i), $i = 1, \ldots, m$, we consider the optimization problem

$$\begin{aligned} \underset{\theta, \xi \geq 0}{\text{minimize}} \;\; & \frac{\lambda}{2}\|\theta\|_2^2 + \sum_{i=1}^m \xi_i \\ \text{subject to} \;\; & f(x_i, y_i; \theta) \leq \min_{y \in \mathcal{Y}} \left(f(x_i, y; \theta) - \Delta(y_i, y)\right) - \xi_i \end{aligned} \tag{5}$$

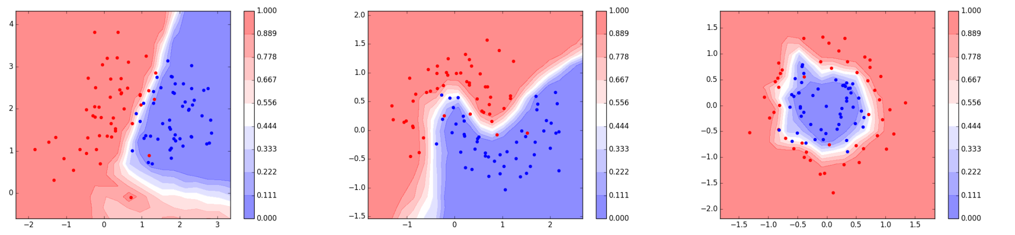
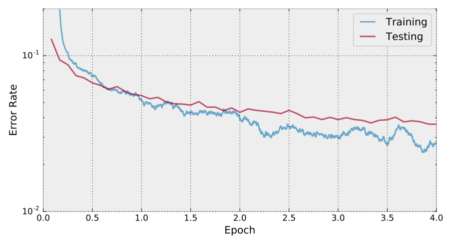
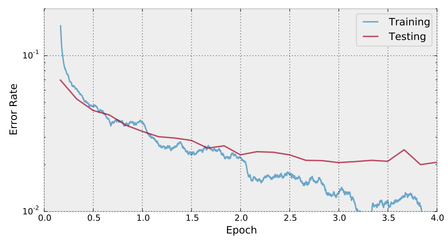


Figure 2: ICNN predictions on 2D classification problems: linear (left), moons (middle) and circle (right). Best viewed in color.



(a) ICNN



(b) FCNN

Figure 3: Training (rolling mean over 100 iterations) and testing (evaluated on full test set) for 600-200 ICNN and FCNN.

where $\Delta(y_i, y)$ is a *margin-scaling* term that requires we satisfy the inequality with some margin for y_i different from y. As a simple example, for multiclass classification tasks where y_i denotes a "one-hot" encoding of examples, we can choose $\mathcal{Y}$ to be the simplex, and $\Delta(y_i, y) = y^T(1-y_i)$. Training requires solving this "loss-augmented" inference problem, which is convex as long as $\Delta(y, y_i)$ is concave, as in standard max-margin structured prediction.

Because the optimization problem (5) is *not* convex in θ, we advocate for solving it via the subgradient method for structured prediction (Ratliff et al., 2007). This algorithm iteratively selects a training example x_i, y_i, then 1) solves the optimization problem

$$y^\star = \operatorname*{argmin}_{y \in \mathcal{Y}} f(x_i, y; \theta) - \Delta(y_i, y) \tag{6}$$

and 2) updates the parameters according to the subgradient

$$\theta := \mathcal{P}_+ \left[\theta - \alpha \left(\lambda \theta + \nabla_\theta f(x_i, y_i, \theta) - \nabla_\theta f(x_i, y^\star; \theta)\right)\right] \tag{7}$$

where P_+ denotes the projection of $W_{1:k-1}^{(z)}$ onto the non-negative orthant. This method can be easily adapted to use mini-batches instead of a single example per subgradient step, and also adapted to alternative optimization methods like AdaGrad (Duchi et al., 2011).

3 Empirical Results

Classifying Synthetic 2D Datasets To understand the classification ICNN's representational power, we trained an ICNN on three synthetic two-dimensional binary classification tasks. The ICNN has two hidden layers, each with 200 units. Figure 2 shows the datasets and the trained model's predicted probability of the "red" class over a grid. The results emphasize that despite their restrictions, fully convex ICNNs are able to learn suitably rich separations of the space.

Classifying MNIST We next compared an ICNN's accuracy to a fully-connected neural network (FCNN) on MNIST (LeCun et al., 1998). The ICNN has two hidden layers with 600 and 200 hidden units, and the fully-connected neural network has the same number of hidden units. We trained using mini-batch sizes of 100 and AdaGrad. As shown in Figure 3 after 4 epochs the ICNN has an error of 3.63%, which is relatively high, but still far lower than a linear classifier (virtually the only other classifier whose predictions are convex in its inputs).

REFERENCES

Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159, 2011.

Yann LeCun, Corinna Cortes, and Christopher JC Burges. The MNIST database of handwritten digits, 1998.

Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and F Huang. A tutorial on energy-based learning. *Predicting structured data*, 1:0, 2006.

Nathan D Ratliff, J Andrew Bagnell, and Martin Zinkevich. (Approximate) subgradient methods for structured prediction. In *International Conference on Artificial Intelligence and Statistics*, pp. 380–387, 2007.

Ben Taskar, Vassil Chatalbashev, Daphne Koller, and Carlos Guestrin. Learning structured prediction models: A large margin approach. In *Proceedings of the 22nd International Conference on Machine Learning*, pp. 896–903. ACM, 2005.

Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484, 2005.