Bregman Neural Networks

Jordan Frecon[1], Gilles Gasso[1], Massimiliano Pontil[2,3], Saverio Salzo[2]

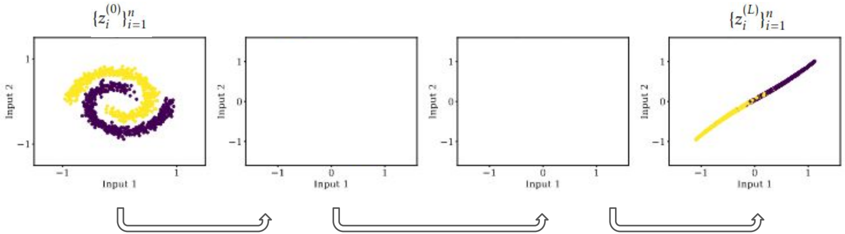
[1] LITIS - INSA Rouen Normandy
[2] CSML - Istituto Italiano di Tecnologia
[3] Dept. of Computer Science - University College London

The Thirty-ninth International Conference on Machine Learning, Baltimore, USA

Context: Representation Learning

Learn a representation linearly separable

Bilevel Framework



$$z_i^{(l+1)} = \underset{z \in \mathbb{R}^d}{\operatorname{argmin}} \; f_l(z, z_i^{(l)}) + D(z, z_i^{(l)}) + g(z)$$

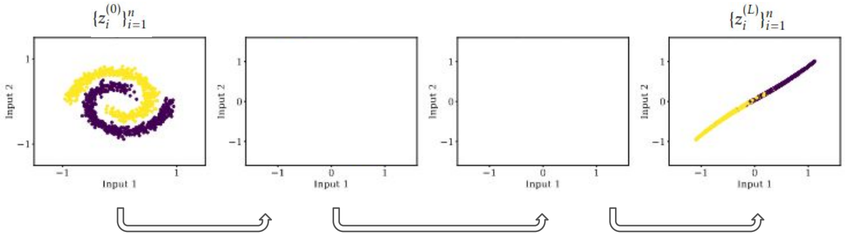
Hyperparameter

- f_l: promotes the next representation $z^{(l+1)}$ depending on the previous one $z^{(l)}$

Fixed

- D: divergence between two successive representations
- g: regularization

Bilevel Framework



$$z_i^{(l+1)} = \operatorname*{argmin}_{z \in \mathbb{R}^d} f_l(z, z_i^{(l)}) + D(z, z_i^{(l)}) + g(z)$$

Multi-layered Bilevel Problem

$$\min_{\psi, \{f_l\}_{l=0}^{L-1}} \sum_{i=1}^{n} \ell(\psi(z_i^{(L)}), y_i) \quad \text{s.c.} \quad \begin{array}{l} z_i^{(0)} = x_i \\ \text{for } l = 0, 1, \ldots L-1 \\ \quad \left\lfloor \, z_i^{(l+1)} = \operatorname*{argmin}_{z \in \mathbb{R}^d} f_l(z, z_i^{(l)}) + D(z, z_i^{(l)}) + g(z) \right. \end{array}$$

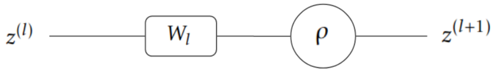
Connections with Neural Networks

Informal Theorem

For f_l bi-linear, D a quadratic distance and g a convex function:

$\Rightarrow$ **Multilayer perceptron**

∇f_l	$\Leftrightarrow$	bias and weight W_l
proximal operator of g	$\Leftrightarrow$	activation function ρ



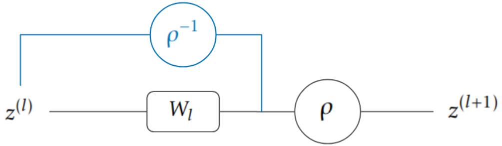
Connections with Neural Networks

Informal Theorem

For f_l bi-linear, $\underline{D \text{ a Bregman distance}}$ and g a convex function:

$\Rightarrow$ **Bregman multilayer perceptron**

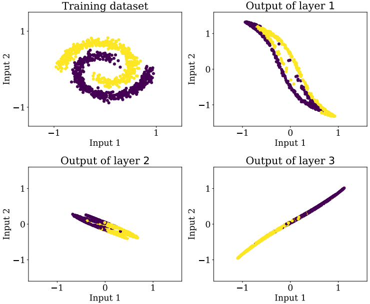
∇f_l	$\Leftrightarrow$	bias and weight W_l
proximal operator of g with Bregman distance	$\Leftrightarrow$	activation function ρ
Bregman distance	$\Leftrightarrow$	residual term



Numerical Experiments on Two-Spiral Dataset

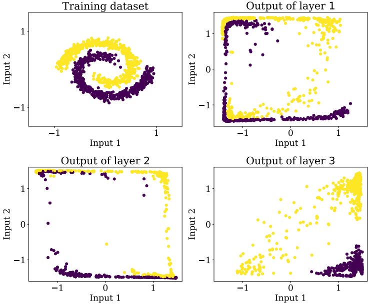
MLP with 3 layers and 2 neurons per layer

Standard MLP



small separation

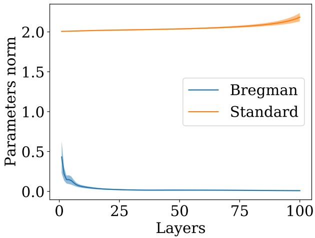
Bregman MLP



higher separation

Numerical Experiments on Two-Spiral Dataset

MLP with 100 layers and 2 neurons per layer



Standard MLP

- increasing impact with #layers
- highly sensible to initialization

Bregman MLP

- decreasing impact with #layers
- slightly sensible to initialization

- $(W_l = 0, b_l = 0) \Rightarrow z^{(l+1)} = z^{(l)}$

Extension to other architectures

ResNet

$$z \leftarrow \mathrm{ReLu}\left(z + \mathrm{Conv2}\Big[\mathrm{ReLu}(\mathrm{Conv1}[z])\Big]\right)$$

BregmanResNet

$$z \leftarrow \rho\left(\rho^{-1}(z) + \mathrm{Conv2}\Big[\rho(\mathrm{Conv1}[z])\Big]\right)$$

Extension to other architectures

ResNet

$$z \leftarrow \mathrm{ReLu}\left(z + \mathrm{Conv2}\left[\mathrm{ReLu}(\mathrm{Conv1}[z])\right]\right)$$

BregmanResNet

$$z \leftarrow \rho\left(\rho^{-1}(z) + \mathrm{Conv2}\left[\rho(\mathrm{Conv1}[z])\right]\right)$$

	Test accuracy	ℓ_∞-Robust accuracy	ℓ_2-Robust accuracy
BregmanResNet20 (atan)	89.11 ($\pm$ 0.20)	33.86 ($\pm$ 0.68)	50.56 ($\pm$ 0.52)
BregmanResNet20 (tanh)	89.28 ($\pm$ 0.28)	35.29 ($\pm$ 1.51)	51.68 ($\pm$ 1.42)
BregmanResNet20 (sigmoid)	89.75 ($\pm$ 0.23)	33.26 ($\pm$ 1.47)	50.40 ($\pm$ 1.46)
BregmanResNet20 (softplus)	**90.82 ($\pm$ 0.12)**	**42.46 ($\pm$ 1.25)**	**53.13 ($\pm$ 1.68)**
ResNet20	90.80 ($\pm$ 0.18)	40.84 ($\pm$ 0.71)	51.65 ($\pm$ 1.32)

Residual networks on CIFAR-10

- Similar test accuracy
- Higher robust accuracies with Bregman variant

Thank You

BregmaNet : Bregman Neural Networks

license MIT release v1.0.0 pypi v1.0.0

BregmaNet is a PyTorch library providing multiple Bregman Neural Networks. To date, implemented models cover Bregman variants of multi-layer perceptrons and various residual networks.

PyTorch library available at `https://github.com/JordanFrecon/bregmanet`