Proximal Neural Networks (PNNs)

J. Hertrich, S. Neumayer, and G. Steidl

TU Berlin, Germany

Proximity Operators

By $\Gamma_0(\mathbb{R}^d)$, we denote the set of proper, convex, lower semi-continuous functions on $\mathbb{R}^d$ mapping into $(-\infty, \infty]$. For $f \in \Gamma_0(\mathbb{R}^d)$ and $T \in \mathbb{R}^{n,d}$, the *proximity operator* $\mathrm{prox}_{f,T} \colon \mathbb{R}^d \to \mathbb{R}^d$ is defined by

$$\mathrm{prox}_{f,T}(x) := \operatorname*{argmin}_{y \in \mathbb{R}^d} \{ \tfrac{1}{2} \|x - y\|_T^2 + f(y) \}.$$

Here $\|\cdot\|_T^2 := \|Tx\|_2^2 / \|T\|^2 + \|P_{\mathcal{N}(T)} x\|_2^2$.
Based on Moreau's characterization of proximity operators, we have shown in [2] that for any $T \in \mathbb{R}^{n,d}$, $b \in \mathbb{R}^n$ and $f \in \Gamma_0(\mathbb{R}^n)$

$$T^\dagger \mathrm{prox}_{f,I_n}(T \cdot + b) = \mathrm{prox}_{g,T} \quad \text{for some } g \in \Gamma_0(R^d).$$

Averaged Operators

An operator $T \colon \mathbb{R}^d \to \mathbb{R}^d$ is *averaged*, if there exists a nonexpansive operator $R \colon \mathbb{R}^d \to \mathbb{R}^d$ such that

$$T = tR + (1-t) I_d \text{ for some } t \in (0,1).$$

Averaged operators with $t = \frac{1}{2}$ are also known as *firmly nonexpansive operators*. Proximity operators are firmly nonexpansive.

Properties of averaged operators:

i) If $T \colon \mathbb{R}^d \to \mathbb{R}^d$ is Lipschitz continuous with Lipschitz constant $L < 1$, then A is averaged for every parameter $t \in [\frac{1}{2}(L+1), 1)$.

ii) The concatenation of K averaged operators T_k with parameters $t_k \in (0,1)$ is an averaged operator with $t \leq \frac{K}{(K-1)+1/\max_k t_k}$.

iii) If $T \colon \mathbb{R}^d \to \mathbb{R}^d$ is an averaged operator with nonempty fixed point set, then the sequence generated by $x^{(r+1)} = Ax^{(r)}$ converges for every starting point $x^{(0)}$ to a fixed point of T.

Plug-and-Play (PnP) Algorithms

Algorithm 1 FBS and FBS-PnP

Initialization: $y^{(0)} \in \mathbb{R}^n$, $\eta \in (0, \frac{2}{L})$
Iterations: For $r = 0, 1, \ldots$

$$y^{(r+1)} = x^{(r)} - \eta \nabla f(x^{(r)})$$
$$x^{(r+1)} = \mathrm{prox}_{\eta g}\left(y^{(r+1)}\right)$$

PnP Step: $x^{(r+1)} = \Psi\left(y^{(r+1)}\right)$

Algorithm 2 ADMM and ADMM-PnP

Initialization: $y^{(0)} \in \mathbb{R}^n$, $p^{(0)} \in \mathbb{R}^n$, $\gamma > 0$
Iterations: For $r = 0, 1, \ldots$

$$x^{(r+1)} = \mathrm{prox}_{\frac{1}{\gamma} f}\left(y^{(r)} - \tfrac{1}{\gamma} p^{(r)}\right)$$
$$y^{(r+1)} = \mathrm{prox}_{\frac{1}{\gamma} g}\left(x^{(r+1)} + \tfrac{1}{\gamma} p^{(r)}\right)$$

PnP Step: $y^{(r+1)} = \Psi\left(x^{(r+1)} + \frac{1}{\gamma} p^{(r)}\right)$

$$p^{(r+1)} = p^{(r)} + \gamma(x^{(r+1)} - y^{(r+1)})$$

Convergence of PnP-Algorithms:

i) Let $f \colon \mathbb{R}^d \to \mathbb{R}$ be differentiable with L-Lipschitz continuous gradient and let $\Psi \colon \mathbb{R}^d \to \mathbb{R}^d$ be averaged. Then, for any $0 < \tau < \frac{2}{L}$, the sequence $\{x^{(r)}\}_r$ generated by FBS-PnP algorithm converges.

ii) Let $f \in \Gamma_0(\mathbb{R}^d)$ and $\Psi \colon \mathbb{R}^n \to \mathbb{R}^n$ be firmly nonexpansive, then the sequence $\{x^{(r)}\}_r$ generated by the ADMM-PnP converges.

PNNs

Let σ_α be a *stable activation functions*, i.e., a function with $\sigma(0) = 0$ and $\sigma = \mathrm{prox}_f$ for some $f \in \Gamma_0(\mathbb{R})$, see [1]. A *proximal neural network* (PNN) is defined as

$$\mathbf{\Phi}(x; u) = T_{K+1} T_K^{\mathrm{T}} \sigma_{\alpha_K}(T_K \cdots T_2^{\mathrm{T}} \sigma_{\alpha_2}(T_2 T_1^{\mathrm{T}} \sigma_{\alpha_1}(T_1 x + b_1) + b_2) \cdots) + b_{K+1},$$

with parameters $u = ((T_k')_{k=1}^{K+1}, (b_k)_{k=1}^{K+1}, (\alpha_k)_{k=1}^{K})$ on the manifold

$$\mathcal{M}_k := \mathrm{St}(d, n_k) \times \mathbb{R}^{n_k} \times \mathbb{R}_{>0}, \quad k = 1, \ldots, K.$$

Here $\mathrm{St}(d,n) := \{T \in \mathbb{R}^{n,d} : T^{\mathrm{T}} T = I_d\}$ denotes the (compact) *Stiefel manifold*. PNNs include OMDSM networks considered in [4]. Learning such networks by *stochastic gradient descent algorithm on the manifold*.

Convolutional PNNs

Use for T_k block matrices with circulant blocks

$$C = \begin{pmatrix} \mathrm{Circ}_m(a^{(1,1)}) & \cdots & \mathrm{Circ}_m(a^{(1,m_2)}) \\ \vdots & & \vdots \\ \mathrm{Circ}_m(a^{(m_1,1)}) & \cdots & \mathrm{Circ}_m(a^{(m_1,m_2)}) \end{pmatrix}$$

Convolutional PNNs with full filters: parameters (T_k, b_k, α_k) on a submanifold of $\mathcal{M}_k$; use basically the same learning algorithm as above.

Convolutional PNNs with sparse filters: parameters are no longer in a manifold; approximate orthogonality property for learning and apply iSPRING algorithm [3].

Numerical Examples

Denoising on piecewise constant signals: Use a PNN with one, two or three layers and $n_1 = n_2 = n_3 = 1024$ neurons in each layer; we use the soft-shrinkage function as activation function and learn the threshold by SGD. As comparison we use Haar frame shrinkage with scale adapted threshold.

Method	PSNR	Loss	Optimal λ
Haar frame	30.59	0.00307	0.0820
One layer PNN	32.50	0.00207	0.0514
Two layer PNN	33.05	0.00186	0.0250
Three layer PNN	33.22	0.00181	0.0164

Stability under adversarial attacks: We train a PNN for MNIST classification. We observe increased stability under adversarial attacks compared to classical classification networks.
PnP-Denoising on piecewise constant signals: We train a PNN for denoising signals with noise level 0.1 and use it within FBS-PnP to denoise signals for various noise levels.

Noise level	0.05	0.1	0.2	0.3	0.5	0.75
PnP step size τ	1.929	0.963	0.479	0.317	0.187	0.122
Noisy signals	31.64	25.61	10.59	16.07	11.63	08.11
Reconstruction PnP	39.05	33.52	28.13	25.05	21.35	18.63
Reconstruction PNN	36.63	33.48	24.53	19.31	13.54	09.37

References

[1] P. L. Combettes and J.-C. Pesquet. Deep neural network structures solving variational inequalities. *Set-Val. Var. Ana.* 2020.

[2] M. Hasannasab, J. Hertrich, S. Neumayer, G. Plonka, S. Setzer, and G. Steidl. Parseval proximal neural networks. *J. Four. Ana.* 2020.

[3] J. Hertrich and G. Steidl. Inertial stochastic PALM and its application for learning Student-t mixture models. *ArXiv* 2020.

[4] L. Huang, X. Liu, B. Lang, A. W. Yu, Y. Wang, and B. Li. Orthogonal weight normalization: Solution to optimization over multiple dependent Stiefel manifolds in deep neural networks. In *32. AAAI Conf.* 2018.

[5] S. Sreehariand, S. V. Venkatakrishnan, and B. Wohlberg. Plug-and-play priors for bright field electron tomography and sparse interpolation. *IEEE Trans. Comput. Imag.* 2016.