Feasibility-based FPNs

Blog Post by Howard Heaton

May 3, 2021

TL;DR – Projection algorithms can now be fused with data-driven regularization to effectively solve inverse problems with many constraints.[1]

The Idea. Inverse problems consist of recovering a signal from a collection of noisy measurements. To find a signal generating provided measurements is known as a feasibility problem. Additional information beyond feasibility must typically be encoded into a model via regularization to ensure accurate and stable recovery of a signal. Hand-chosen analytic regularization has limited effectiveness due its inability to leverage large amounts of available data. To this end, feasibility-based fixed point networks (F-FPNs) were recently proposed to fuse data-driven regularization and feasibility in a theoretically sound manner. Each F-FPN defines a collection of operators, each of which is the composition of a feasibility-based operator and a data-driven regularization operator. Fixed point iteration is used to compute fixed points of these operators. Weights of the regularization operator are tuned to ensure the fixed points closely represent available data. This results in data-driven algorithms that inherit desirable properties from classic approaches.[2]

[1] Based on this paper by Heaton, Wu Fung, Gibali, Yin.

[2] In this context, classic typically refers to methods that were designed by hand (*e.g.* gradient descent on a known function) rather than derived from a black-box approach (*e.g.* a fully connect neural network).

The Setup Suppose we wish to recover a signal $u_d^\star \in \mathbb{R}^n$ using measurement data $d \in \mathbb{R}^m$. For some matrix $A \in \mathbb{R}^{m\times n}$, the data d relates to $u_d^\star$ via[3]

$$d = Au_d^\star + \varepsilon, \tag{1}$$

where ε is some unknown noise. For each element d_i, we let $\mathcal{C}_{d,i}$ denote the set of signals satifying the d_i-th measurement (called a *hyperplane*), *i.e.*

$$\mathcal{C}_{d,i} \triangleq \left\{u : \left\langle a^i, u\right\rangle = d_i\right\}, \quad \text{for all } i \in [m]. \tag{2}$$

We let $P_{\mathcal{C}_{d,i}}$ denote the orthogonal projection onto $\mathcal{C}_{d,i}$.[4] If there is no noise (*i.e.* $\varepsilon = 0$), then $u_d^\star$ is in the intersection $\mathcal{C}_d$ of all of hyperplanes; namely,[5]

$$u_d^\star \in \mathcal{C}_d \triangleq \bigcap_{i=1}^m \mathcal{C}_{d,\ell} = \bigcap_{i=1}^m \text{fix}\left(P_{\mathcal{C}_{d,i}}\right). \tag{3}$$

In practice, estimating $u_d^\star$ by simply finding a point in $\mathcal{C}_d$ is problematic for two reasons. When A is short and fat (*i.e.* $m < n$), there are infinitely many points in $\mathcal{C}_d$, leaving it unclear how to identify $u_d^\star$. To make matters worse, when noise is present (*i.e.* $\varepsilon \neq 0$), the set $\mathcal{C}_d$ is often the empty set! That is, we can't even hope to find $u_d^\star$ by only modeling with the intersection set $\mathcal{C}_d$.[6]

[3] This section presents a special case of the more general/abstract setting of the F-FPN paper.

[4] The projection $P_{\mathcal{C}}$ onto a closed and convex set $\mathcal{C} \subseteq \mathbb{R}^n$ is defined by

$$P_{\mathcal{C}}(x) \triangleq \operatorname*{arg\,min}_{z\in\mathcal{C}} \|z-x\|^2.$$

[5] For an operator T, we set

$$\text{fix}(T) \triangleq \{u : T(u) = u\}.$$

[6] This is one reason why it is common in practice to use sums of error terms (*e.g.* least squares) or early stopping to avoid overfitting to noise.

The F-FPN paper presents a way to augment the task of finding $u_d^\star \in \mathcal{C}_d$ for practical settings. The key trick is to introduce regularization into each of the projection terms, resulting in an operation resembling projected gradient. That is, for a regularization operator R_Θ (see Figure 1), we assume

$$u_d^\star \in \mathcal{C}_{\Theta,d} \triangleq \bigcap_{i=1}^m \text{fix}\left(P_{\mathcal{C}_{d,i}} \circ R_\Theta\right). \tag{4}$$

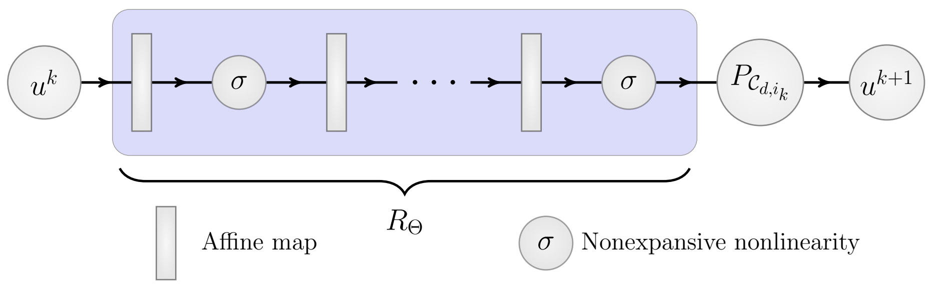

Figure 1: Diagram of feed forward updates inside F-FPN (see Algorithm 2). The operator R_Θ can be any 1-Lipschitz neural network with $\mathbb{R}^n$ as both the input and output spaces. Typical examples of nonexpansive nonlinearities σ include ReLUs (*i.e.* the projection onto the set $\mathbb{R}^n_{\geq 0}$).

If R_Θ is the identity, then $\mathcal{C}_{\Theta,d} = \mathcal{C}_d$. If R_Θ is well-chosen, then $\mathcal{C}_{\Theta,d}$ has one element: the signal $u_d^\star$. Assuming $\mathcal{C}_{\Theta,d}$ is a singleton set,[7] an F-FPN is given by

$$\mathcal{M}_\Theta(d) \triangleq \tilde{u}_d, \quad \text{where} \quad \tilde{u}_d \in \mathcal{C}_{\Theta,d}. \tag{5}$$

Each F-FPN $\mathcal{M}_\Theta$ can be evaluated using Algorithm 2.

[7] The set $\mathcal{C}_{\Theta,d}$ likely contains multiple elements in practice; however, since we start with the same estimate u^1, the limit of Algorithm 2 will be consistent.

1: $\mathcal{M}_\Theta(d)$:
2: $u^1 \leftarrow \tilde{u}$
3: **while** $\|u^{k+1} - u^k\| > \varepsilon$
4: $i_k \leftarrow k \bmod(m) + 1$
5: $u^{k+1} \leftarrow (P_{\mathcal{C}_{d,i_k}} \circ R_\Theta)(u^k)$
6: $k \leftarrow k + 1$
7: **return** u^k

Algorithm 2: F-FPN starts with an arbitrary initialization $\tilde{u}$ in Line 2 and repeatedly applies projections $P_{\mathcal{C}_{i_k,d}}$ and regularization R_Θ until a convergence.

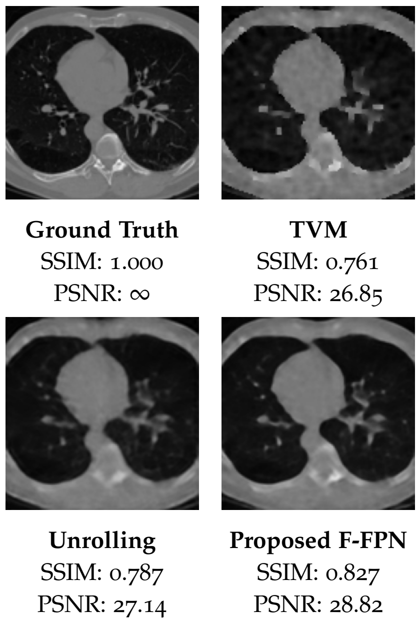

Figure 3: Zoomed-in snippet of LoDoPab CT on test data, using TV minimization (TVM), 12 step unrolled network, and F-FPN.

REGULARIZATION The regularization operator R_Θ can be chosen to be anything that is 1-Lipschitz.[8] In this work, the authors focused on problems where $u_d^\star$ is an image. Consequently, R_Θ was chosen to be a ResNet with convolutions. Training data are provided in pairs $(d, u_d^\star)$ with measurements d drawn from a distribution $\mathcal{D}$. The weights Θ are tuned to solve the training problem

$$\min_\Theta \mathbb{E}_{d\sim\mathcal{D}} \left[\|\mathcal{M}_\Theta(d) - u_d^\star\|^2 \right]. \tag{6}$$

[8] A mapping T is 1-Lipschitz provided $\|T(u) - T(v)\| \leq \|u - v\|$ for all u and v.

EXPERIMENTS The experiments presented in the F-FPN paper were CT image reconstruction problems. Snippets of some results are shown in Figures 3 and 4 on test data. The results with F-FPNs are surprisingly good, particularly since $A \in \mathbb{R}^{5490\times16384}$ is short and fat and d is noisy. The F-FPN images here were generated using 100-150 steps of Algorithm 2, which amounted to successively evaluating 1000+ blocks of convolutions (*i.e.* the F-FPN $\mathcal{M}_\Theta$ is a *deep* network). Without memory efficient backprop (*e.g.* see the FPN blog), implicit depth models would effectively be impossible on a standard GPU. To illustrate the advantage of this depth and backprop, the authors present an unrolled version of F-FPNs with traditional backprop, fixed to ≤ 20 steps of Algorithm 2 (due to memory limits). Unsurprisingly, naïve unrolling results in lower quality results compared to the proposed F-FPNs. Also, R_Θ needed only 96K parameters to generate these results (small be ML standards).

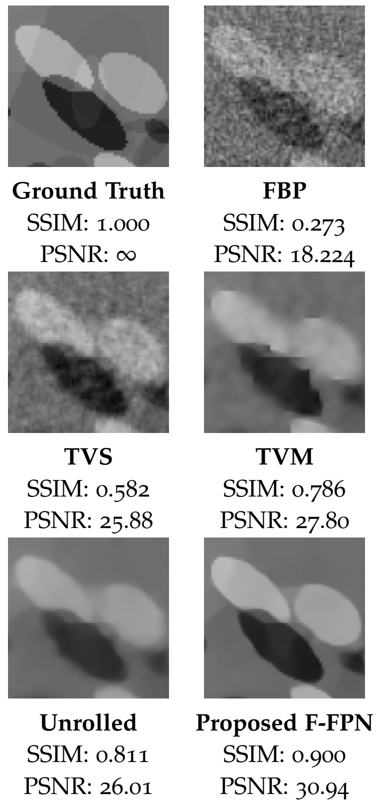

Figure 4: Zoomed-in snippet of ellipses CT on test data, using filtered backprojection (FBP), total variation superiorization/minimization (TVS/TVM), 20 step unrolled network, and F-FPN.

CONCLUSIONS Projection-based algorithms have been used with success in many applications with massive-scale constraints. However, projection methods have, until now, not leveraged big data. Fusing in data-driven regularization improves reconstruction quality without sacrificing theoretical rigor. Moreover, F-FPNs are memory efficient and scalable to massive problems. This illustrates a mutually beneficial relationship between classic projection algorithms and deep learning, which will be further explored.

Last Updated: May 3, 2021