

Lecture 26: Plug-and-Play — Motivation; Plug-In Denoisers

Charles A. Bouman

Purdue University

Monday, October 26, 2026

Outline

- 1 Regularized inversion, $x^* = \arg \min_x \{f(x) + h(x)\}$, and its limitations
- 2 Objectives versus actions; the sensor and prior manifolds
- 3 The proximal maps $F(x)$ and $H(x)$
- 4 PnP-ADMM, and why it is called “plug-and-play”
- 5 Designing the plug-in denoiser: agents, the virtual model, and supervised training
- 6 Section 10.2: are plug-in denoisers proximal maps? Moreau’s Theorem

Core idea

The proximal map converts an objective into an action. Once ADMM is written purely in terms of the two actions F and H , we can *plug in* a general denoiser in place of H — and the algorithm is then no longer solving an optimization problem.

Today's Big Ideas

- 1 Regularized inversion estimates the unknown by minimizing a data-fitting term plus a regularizing term, and it is limited because the two terms must be minimized together and because both models must be written as cost functions.
- 2 The proximal map converts a cost function into an action, which is an operator that takes an image in and returns an improved image.
- 3 PnP-ADMM alternates the action of the forward model with the action of the prior model, and plugging a general operator in place of either action means the algorithm is no longer minimizing a cost.
- 4 The action that replaces the prior model has the form of a denoiser for additive white Gaussian noise, so it can be trained on pairs of clean images and synthetically noised copies of them.
- 5 A trained denoiser is not likely to be a proximal map, and the larger class of agents that are not proximal maps offers the potential for better reconstructions.

Why this chapter

Recent progress in machine learning and DNNs has made it clear that these emerging methods can represent very complex phenomena. This raises the question of how to best use them in Computational Imaging applications.

- One obvious approach (which we will not treat here) is to simply learn the relationship between the data, y , and the unknown, x , from training data.
- In real applications this direct approach is often not desirable, because we would like to incorporate detailed physical models, particularly sensor models that we would rather not replace with machine learning algorithms.
- Direct learning methods also tend to be one-size-fits-all, without the modular structure that makes engineering designs flexible and maintainable.

Goal of Plug-and-Play (PnP)

Fuse the cost minimization models used up to this point in the text with **action-based** models, such as DNNs, that are designed to generate outputs from inputs.

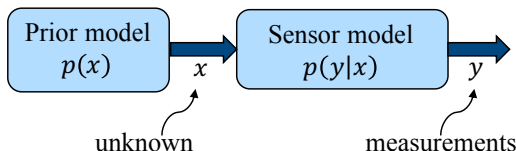
Section 10.1: regularized inversion

The MAP estimate and the regularized ML estimate are general methods for solving inverse problems that can be expressed in the form

$$x^* = \arg \min_x \{f(x) + h(x)\} , \quad (10.1)$$

where x^* emphasizes its role as the solution to an optimization problem.

- For the MAP estimate, $f(x) = -\log p(y|x) + \text{const}$ and $h(x) = -\log p(x) + \text{const}$.
- More generally, $f(x)$ can be any data-fitting expression such as total squared error, and $h(x)$ can be any stabilizing function that penalizes irregular behavior in the solution.



Inverse problems represent a broad class of important problems in which the goal is to estimate an unknown or latent quantity, x , from available measurements, y .

Two severe limitations of this framework

This optimization framework for regularized inversion has served us well, but it has some severe limitations.

First: it is not modular

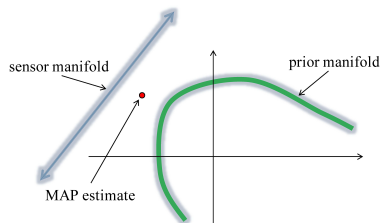
Both terms must be optimized together. This can be challenging in real applications for which the forward and prior models might represent tens of thousands of lines of code!

Second: it is objective-based

The solution of (10.1) is purely based on the **objective** of minimizing the cost functions $f(x)$ and $h(x)$, but many powerful techniques are based on **actions** in the form of machine learning functions, $F(x)$, that take an input and transform it to some desired output.

From Chapter 9: the splitting methods of constrained optimization provide some useful methods for breaking large problems into smaller manageable pieces; we build on these here.

The sensor manifold and the prior manifold



The regularized inverse balances fitting the sensor data against being a plausible solution. For a linear sensor with AWGN, this is finding a point close to both the sensor and the image manifold, with

$$f(x) = \frac{1}{2} \|y - Ax\|^2.$$

- The prior manifold is almost always very complex and nonlinear in structure.
- Intuitively, the prior manifold is always nonlinear because real images have enormously intricate structure.

Consequently, machine learning methods such as DNNs are serving an increasingly important role in accurately capturing the complex structure in the prior model's thin manifold.

Section 10.1.1: the proximal maps of f and h

To apply ADMM and variable splitting to (10.1), we first introduce the proximal maps associated with $f(x)$ and $h(x)$:

$$F(x) = \arg \min_v \left\{ f(v) + \frac{1}{2\sigma^2} \|v - x\|^2 \right\}, \quad (10.2)$$

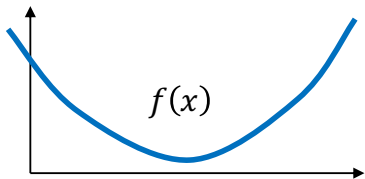
$$H(x) = \arg \min_v \left\{ h(v) + \frac{1}{2\sigma^2} \|v - x\|^2 \right\}, \quad (10.3)$$

where σ^2 is a selectable parameter.

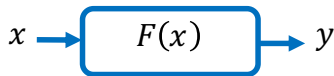
The technical setting

In applications $f(v)$ and $h(v)$ are often just well-behaved continuous functions. But to prove theorems about convergence to a unique global minimum, we usually need stronger assumptions. So it is common to assume $f(v)$ and $h(v)$ are **proper closed convex functions** taking values on the **extended real line**, $\mathbb{R} \cup \{\infty\}$, which adds the value ∞ to the reals. Such a function is sometimes called an **extended convex function**. This is useful since the value ∞ provides a simple mechanism of imposing constraints in optimization problems.

Objectives versus actions



(a) Model of an Objective



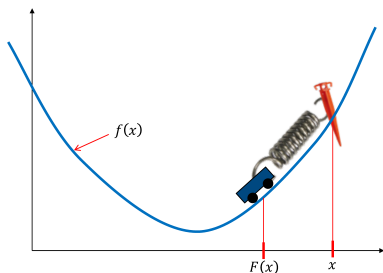
(b) Model of an Action

- $f(x)$ in (a) is an **objective**: the goal is to change x in a way that reduces the value of the function.
- $F(x)$ in (b) is an **action**: when it is applied to x it creates the new value, $F(x)$.

Converting objectives to actions

The proximal map is a very subtle but powerful idea because it provides a mathematical machine for converting objectives to actions. The proximal map $F(x)$ represents an **action** we take that moves us closer to the minimum of $f(x)$.

Interpretation of the proximal map



(c) Interpretation of Proximal Map

- Intuitively, the proximal map is the resting position of a cart that is attached to a spring that is anchored at a point x .
- Gravity causes the cart to move down the function, but the spring restrains the cart and keeps it from moving to the bottom.
- So the objective function, $f(x)$, creates an action operator, $F(x)$, that results in the movement of the cart.

A central question of this chapter is how to fuse models that are represented as a heterogeneous combination of objectives and actions.

The PnP-ADMM algorithm

We use the proximal maps to split the optimization problem into two parts and solve the parts separately using ADMM.

PnP-ADMM Algorithm

Initialize $u \leftarrow \mathbf{0}$ and $v \leftarrow \mathbf{0}$

Until Converged {

$$x \leftarrow F(v - u)$$

$$v \leftarrow H(x + u)$$

$$u \leftarrow u + (x - v)$$

}

Return(x)

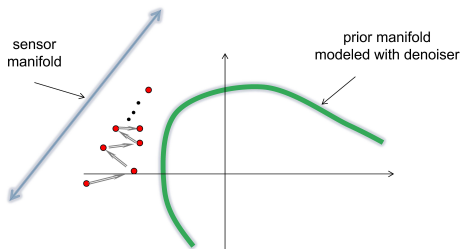
- When $F(x)$ and $H(x)$ are the proximal maps (10.2) and (10.3), this is **exactly the ADMM algorithm** of the previous chapter.

Why “plug-and-play”

The power of PnP comes when we replace $F(x)$ and $H(x)$ with more general actions. This is a huge conceptual step because when $F(x)$ and $H(x)$ are no longer proximal maps, then **this algorithm is no longer solving an optimization problem**. We have simply plugged in a different action where the proximal map should have been.

The PnP prior

- The PnP prior replaces $H(x)$ with an action designed or trained to remove AWGN with variance σ^2 .
- The optimized denoiser then serves as a much better prior model, resulting in a much more effective reconstruction algorithm.
- The power of PnP is that it can use advanced machine learning methods, such as DNNs, that much more accurately model real images.



Each iteration applies an action F that moves the solution closer to the sensor manifold, and an action H that moves the solution closer to the data manifold. The hope is that these iterations will eventually converge to a solution that balances these two actions.

Section 10.1.2: what should we plug in?

If we are not going to plug in the proximal map, then what new operator should we plug in instead?

- Our goal is to replace the proximal maps in the ADMM algorithm with more general operations that represent more useful actions.
- From now on we refer to these operations as **agents**, since they have the anthropomorphic interpretation of imposing their will on the solution.

Take a closer look at the proximal map $H(x)$:

$$H(x) = \arg \min_v \left\{ \frac{1}{2\sigma^2} \|x - v\|^2 + h(v) \right\} . \quad (10.4)$$

From this, we can see that $H(x)$ has the form of the **MAP estimate of an image corrupted by AWGN with variance σ^2** .

The virtual model

To make this observation more clear, consider a **virtual model** of the form

$$X = V + W ,$$

where V is an image that is typical of what we expect in our reconstruction problem and W is synthetically generated AWGN with variance σ^2 .

If we assume a prior model for the image of the form $V_n \sim \frac{1}{z} \exp\{-h(v)\}$, then the log likelihood of the observations is

$$\log p(x|v) = -\frac{1}{2\sigma^2} \|x - v\|^2 - \log z ,$$

and the MAP estimate of V given X is **exactly the proximal map** $H(X)$ of (10.4).

Generating the training data

Remember that X and V are not actually images in our original reconstruction problem. Instead, they are virtual images that we can imagine exist. However, just because they were born as virtual images doesn't mean we can't generate them synthetically.

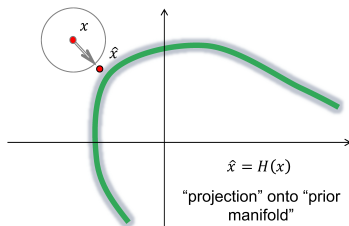
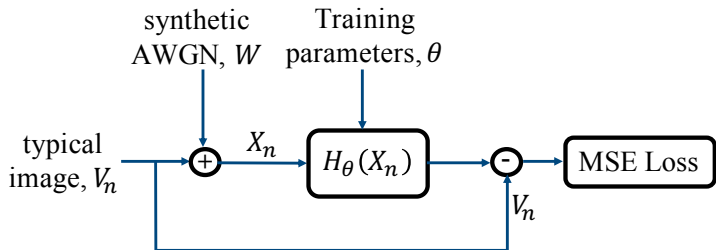
- Take examples of real images from our application, V_n , for $n = 1, \dots, N$, and add Gaussian white noise with variance σ^2 to generate X_n .
- The resulting image pairs, (V_n, X_n) , can then be used for **supervised training** of the function H .
- The inference algorithm can be trained using the MSE loss function

$$L(\theta) = \sum_n \|V_n - H_\theta(X_n)\|^2.$$

Footnote in the text

Technically, this loss function results in the conditional mean rather than the MAP estimate, but in practice, this loss function is very effective for training the PnP denoiser.

Self-training of the PnP prior



- The agent $H_\theta(x)$ is trained to denoise X_n in a manner that minimizes MSE loss relative to V_n .
- So $H_\theta(x)$ can be thought of as an image denoiser, since its output is an MMSE estimate of the noiseless image, V_n .
- The agent behaves similarly to a proximal map: given an input image X_n , it maps the image to a new point that is closer to the prior manifold.

Section 10.2: the questions this raises

We have replaced the proximal map $H(x)$ with a denoiser designed to effectively remove AWGN. This turns out to work amazingly well — but what does it mean?

- Does the PnP-ADMM algorithm converge? (An anonymous reviewer aptly renamed this algorithm “plug-and-pray.”)
- And if it does converge, what might it converge to?

Two variations of the convergence question

When the agents F and H are proximal maps, the converged answer represents the solution to the optimization problem of (10.1). In this case, proving convergence of PnP is equivalent to proving convergence of ADMM from Chapter 9. We never provided that proof, but we will provide it in this chapter.

- The entire concept of PnP is to replace the proximal denoising operator H with a more general denoising algorithm designed with nonparametric machine learning methods.
- So if we drop in this new H , will it still be a proximal map? In most cases, the answer will turn out to be a definitive “no.”

Moreau's Theorem

To answer this question we use pioneering work first done by Moreau in the 1960s.

Theorem: Moreau's Theorem

Let $H : \mathbb{R}^N \rightarrow \mathbb{R}^N$ be a function. Then H is a proximal map of a proper closed convex function if and only if

- H is **nonexpansive** and
- H is **conservative**.

The essential ideas: H is nonexpansive if $\forall x \in \mathbb{R}^N$,

$$\|H(x) - H(y)\| \leq \|x - y\| ,$$

and it is conservative if there exists a $\phi : \mathbb{R}^N \rightarrow \mathbb{R}$ such that $\forall x \in \mathbb{R}^N$,

$$H(x) = \nabla \phi(x) .$$

The self-adjoint gradient condition

So when is a particular H a proximal map? From Moreau's Theorem, it must be conservative. In addition, H must also have a **self-adjoint gradient**:

$$\nabla H(x) = [\nabla H(x)]^t .$$

Example: a linear function

Consider $H(x) = Ax$ for a square matrix A . Then $\nabla H(x) = A$, so the self-adjoint condition is that the matrix is **symmetric**, that is, $A^t = A$.

- Generally speaking, this is a strong constraint that you would not necessarily expect to hold for many functions.
- This is analogous to the classic problem in electromagnetics of determining when a particular vector field can possibly be a valid electric field. The answer is that such a potential field only exists when the electric field has “zero curl,” which is exactly the statement that the gradient of the field is self-adjoint.

Summary

- Regularized inversion: $x^* = \arg \min_x \{f(x) + h(x)\}$; MAP is $f = -\log p(y|x)$, $h = -\log p(x)$. Not modular; objective-based.
- The proximal maps F and H of (10.2)–(10.3) convert objectives into actions; σ^2 is a selectable parameter.
- PnP-ADMM: $x \leftarrow F(v - u)$, $v \leftarrow H(x + u)$, $u \leftarrow u + (x - v)$. With proximal maps this is exactly ADMM; with general actions it is no longer solving an optimization problem.
- $H(x)$ has the form of a MAP denoiser for AWGN of variance σ^2 , so it can be trained from synthetic pairs (V_n, X_n) using $L(\theta) = \sum_n \|V_n - H_\theta(X_n)\|^2$.
- Moreau: H is a proximal map if and only if it is nonexpansive and conservative; a conservative H must also have a self-adjoint gradient, $\nabla H(x) = [\nabla H(x)]^t$.

The important point

Many interesting actions $H(x)$ may not have self-adjoint gradients, so they may not be proximal maps. The much richer class of agents not limited to proximal maps offers the potential for much better reconstruction results. A DNN trained for optimal image denoising is **not likely to be a proximal map, but it could serve as an excellent prior model in a PnP framework**.

Reading: Sections 10.1–10.2. Next lecture: Section 10.3, Consensus Equilibrium for Two Models.