Lecture # 3: Plug & play methods

DEep Learning for Image REstoration and Synthesis

Andrés ALMANSA   Saïd LADJAL   Alasdair NEWSON

Cours de Master M2 MVA
`http://delires.wp.imt.fr`

February 7, 2019

Motivation

Image acquisition model

- ideal image $u : \mathbb{R}^2 \to [0, \infty)$
- Measurements $\tilde{u} : \mathbb{Z}^2 \to [0, 255] \cap \mathbb{Z}$

$$\tilde{u} = g(\Delta_{\mathbb{Z}^2}((u \circ \phi) * h + n))$$

- Geometric deformation ϕ
- blur kernel h
- contrast change g

Simplified model

- $\mathbf{x} = \Delta_{\mathbb{Z}^2 \cap [0,N)^2}(u * \mathrm{sinc})$ (ideal discrete image $\mathbf{x} \in \mathbb{R}^{N^2}$)
- $\tilde{\mathbf{x}} = A(\mathbf{x}) + \mathbf{n}$ (degraded measurements)

Recall (previous course)

Simplified model

- $\mathbf{x} = \Delta_{\mathbb{Z}^2 \cap [0,\sqrt{N})^2}(u * \mathrm{sinc})$ (ideal discrete image $\mathbf{x} \in \mathbb{R}^N$)
- $\tilde{\mathbf{x}} = A(\mathbf{x}) + \mathbf{n}$ (degraded measurements)

Ill-posed $A \Rightarrow$ need for regularization (prior knowledge)

- risk minimization (MMSE): $\min_{\mathbf{x}} \mathbb{E}\left[\|X - \mathbf{x}\|^2 \,\middle|\, \tilde{X} = \tilde{\mathbf{x}} \right]$
- posterior maximisation (MAP) $\max_{\mathbf{x}} \mathbb{P}\left[X = \mathbf{x} \,\middle|\, \tilde{X} = \tilde{\mathbf{x}} \right]$

Special case $\mathbf{n} \sim N(0, \sigma^2)$ and $\mathbf{x} \sim N(\mu, \Sigma)$ are Gaussian

Both MMSE & MAP lead to the same closed form (Wiener filter):
$\hat{x} = (A^* A + \sigma^2 \Sigma^{-1})^{-1} (A^* \tilde{\mathbf{x}} + \sigma^2 \Sigma^{-1} \mu)$

WANING!! This is a very special case !!

Neural Networks for inverse problems: Two paradigms

- Learning-based approach : find a sufficient number of image pairs $(\mathbf{x}^i, \tilde{\mathbf{x}}^i)$ and train a neural network f_θ to invert A by minimizing the empirical risk $\sum_i \|f_\theta(\tilde{\mathbf{x}}^i) - \mathbf{x}^i\|_2^2$
  - ✓ no need to model A, $\mathbf{n}$ nor prior for $\mathbf{x}$
  - ✗ needs retraining if A or $\mathbf{n}$ change
- Bayesian approach : Model separately
  1. conditional probability $\mathbb{P}\left[\tilde{X} = \tilde{\mathbf{x}} \,\middle|\, X = \mathbf{x}\right]$ (using physical model, calibration)
  2. prior model $\mathbb{P}\left[X = \mathbf{x}\right]$ (through NN learning)
  3. Use Bayes theorem to estimate $\mathbf{x}$ via MAP or MMSE

Example of MAP estimation :

- Observation model:
$\mathbb{P}\left[\tilde{X} = \tilde{\mathbf{x}} \,\middle|\, X = \mathbf{x}\right] = \mathbb{P}\left[N = (\tilde{\mathbf{x}} - A(\mathbf{x}))\right] = Ce^{-\frac{1}{2\sigma^2}\|\tilde{\mathbf{x}}-A(\mathbf{x})\|_2^2}$
- Prior model $\mathbb{P}\left[X = \mathbf{x}\right] = C'e^{-\lambda R(\mathbf{x})}$
- Posterior: $-\log \mathbb{P}\left[X = \mathbf{x} \,\middle|\, \tilde{X} = \tilde{\mathbf{x}}\right] = E(\mathbf{x}) =$
$\underbrace{\frac{1}{2\sigma^2}\|\tilde{\mathbf{x}} - A(\mathbf{x})\|_2^2}_{F(\mathbf{x})} + \lambda R(\mathbf{x}) + C''$

Choosing a prior for the clean image x

- Tikhonov regularization: *(convex, smooth)*

$$R(\mathbf{x}) = \sum_i \|\nabla x_i\|_2^2$$

- Total Variation regularization: *(convex, non-smooth)*

$$R(\mathbf{x}) = \sum_i \|\nabla x_i\|_2$$

- Wavelet shrinkage: *(convex, non-smooth)*

$$R(\mathbf{x}) = \|W\mathbf{x}\|_1$$

- Patch based regularization (EPLL): *(non-convex, smooth)*

Choosing a prior for the clean image $\mathbf{x}$

- Tikhonov regularization: *(convex, smooth)*

$$R(\mathbf{x}) = \sum_i \|\nabla x_i\|_2^2$$

- Total Variation regularization: *(convex, non-smooth)*

$$R(\mathbf{x}) = \sum_i \|\nabla x_i\|_2$$

- Wavelet shrinkage: *(convex, non-smooth)*

$$R(\mathbf{x}) = \|W\mathbf{x}\|_1$$

- Patch based regularization (EPLL): *(non-convex, smooth)*
- Neural network to learn $R(\mathbf{x})$?
  - ▲ difficult to train !
  - ▲ properties of R ? (we need to minimise $F(\mathbf{x}) + \lambda R(\mathbf{x})$!!)
- Indirect (Plug & Play) approach :
  - Train a neural network to solve a simpler (denoising) problem:

Choosing a prior for the clean image $\mathbf{x}$

- Indirect (Plug & Play) approach :
  - Train a neural network to solve a simpler (denoising) problem:

$$D_\sigma : \tilde{\mathbf{x}} \mapsto \arg\min_{\mathbf{x}} \frac{1}{2\sigma^2} \|\mathbf{x} - \tilde{\mathbf{x}}\|_2^2 + R(\mathbf{x})$$

  - $D_\sigma = \mathrm{prox}_{\sigma^2 R}$ is the proximal operator of the regularizer
  - Use D_σ to regularize the original problem

Plug & Play approach

A myriad of solutions have been proposed:

- RED - Regularization by Denoising [Romano, Elad & Milanfar 2017; Reehorst & Schnitter 2018]
- HQS - Half Quadratic Splitting [Geman & Yang 2002; Zoran & Weiss 2012]
- ADMM - Alternated Direction Method of Multipliers [Boyd 2010; Chan et al 2017]
- Chambolle-Pock method [Chambolle & Pock 2011-2016, Meinhardt et al 2017]

Questions

- Does the scheme converge?
- Does there exist a regularizer R such that $\mathrm{prox}_{\sigma^2 R} = D_\sigma$?

RED - Regularisation by Denoising

RED constructs an explicit regularizer from a denoiser:

$$R_{RED}(\mathbf{x}) = \frac{1}{2}\langle \mathbf{x}, \mathbf{x} - D_\sigma(\mathbf{x})\rangle$$

Theorem (RED's gradient [Reehorst & Schnitter 2018])

If the denoiser D_σ :

- *is locally homogeneous (i.e. $(1+\varepsilon)D_\sigma(\mathbf{x}) = D_\sigma((1+\varepsilon)\mathbf{x})$) and*
- *has symmetric Jacobian*

then R_{RED}'s gradient is writes $\nabla R_{RED}(\mathbf{x}) = \mathbf{x} - D_\sigma(\mathbf{x})$

RED - Regularisation by Denoising

$$R_{RED}(\mathbf{x}) = \frac{1}{2}\langle \mathbf{x}, \mathbf{x} - D_\sigma(\mathbf{x})\rangle$$

Theorem (RED's gradient [Reehorst & Schnitter 2018])

If the denoiser D_σ :

- *is locally homogeneous (i.e. $(1+\varepsilon)D_\sigma(\mathbf{x}) = D_\sigma((1+\varepsilon)\mathbf{x})$) and*
- *has symmetric Jacobian*

then R_{RED}'s gradient is writes $\nabla R_{RED}(\mathbf{x}) = \mathbf{x} - D_\sigma(\mathbf{x})$

Collorary A gradient descent scheme (without splitting) is easy to implement:

$$\nabla E(\mathbf{x}) = (\nabla F + Id - D_\sigma)(\mathbf{x})$$

RED - Regularisation by Denoising

$$R_{RED}(\mathbf{x}) = \frac{1}{2}\langle \mathbf{x}, \mathbf{x} - D_\sigma(\mathbf{x})\rangle$$

Theorem (RED's gradient [Reehorst & Schnitter 2018])

If the denoiser D_σ :

- *is locally homogeneous (i.e. $(1+\varepsilon)D_\sigma(\mathbf{x}) = D_\sigma((1+\varepsilon)\mathbf{x})$) and*
- *has symmetric Jacobian*

then R_{RED}'s gradient is writes $\nabla R_{RED}(\mathbf{x}) = \mathbf{x} - D_\sigma(\mathbf{x})$

Lemma 1: If the denoiser D is locally homogeneous then $[J_R(\mathbf{x})]\mathbf{x} = D(\mathbf{x})$

Lemma 2: R_{RED}'s gradient is $\nabla R(\mathbf{x}) = \mathbf{x} - 0.5D_\sigma(\mathbf{x}) - 0.5[J_R(\mathbf{x})]^T\mathbf{x}$

Lemma 3: If the denoiser D is locally homogeneous and has symmetric Jacobian then $\nabla R_{RED}(\mathbf{x}) = \mathbf{x} - D(\mathbf{x})$

RED - Regularisation by Denoising

$$R_{RED}(\mathbf{x}) = \frac{1}{2}\langle \mathbf{x}, \mathbf{x} - D_\sigma(\mathbf{x})\rangle$$

Theorem (RED's gradient [Reehorst & Schnitter 2018])

If the denoiser D_σ :

- *is locally homogeneous (i.e. $(1+\varepsilon)D_\sigma(\mathbf{x}) = D_\sigma((1+\varepsilon)\mathbf{x})$) and*
- *has symmetric Jacobian*

then R_{RED}'s gradient is writes $\nabla R_{RED}(\mathbf{x}) = \mathbf{x} - D_\sigma(\mathbf{x})$

Reality check

$$e_f^J = \frac{\|J_f - J_f^T\|_F^2}{\|J_f\|_F^2}$$

	TDT	MF	NLM	BM3D	TNRD	DnCNN
$e_f^J(\boldsymbol{x})$	5.36e-21	1.50	0.250	1.22	0.0378	0.0172

TABLE I
AVERAGE JACOBIAN-SYMMETRY ERROR ON 16×16 IMAGES

HQS - Half Quadratic Splitting

Instead of minimizing the original problem

$$E(\mathbf{x}) = F(\mathbf{x}) + \lambda R(\mathbf{x})$$

we introduce an auxiliary variable $\mathbf{v}$ and we aim at solving the equivalent *constrained minimization problem* :

$$\min_{\mathbf{x},\mathbf{v}} F(\mathbf{x}) + \lambda R(\mathbf{v}) \quad \text{under the constraint } \mathbf{x} = \mathbf{v}$$

The constraint can be added back to the energy (preceded by a Lagrange multiplier β)

$$E_1(\mathbf{x}, \mathbf{v}, \beta) = F(\mathbf{x}) + \beta\|\mathbf{x} - \mathbf{v}\|^2 + \lambda R(\mathbf{v})$$

We know that for β large enough joint minimization of E_1 is equivalent to minimizing E.

HQS - Half Quadratic Splitting

$$E_1(\mathbf{x}, \mathbf{v}, \beta) = F(\mathbf{x}) + \beta\|\mathbf{x} - \mathbf{v}\|^2 + \lambda R(\mathbf{v})$$

To ensure that β is large enough we can use a *continuation scheme*:

$$\arg\min_{\mathbf{x}} E(\mathbf{x}) = \lim_{\beta\to\infty} \arg\min_{\mathbf{x}} \min_{\mathbf{v}} E_1(\mathbf{x}, \mathbf{v}, \beta)$$

Which inspires this alternating a minimization algorithm:

1. Initialization: $\mathbf{x}^0, \mathbf{v}^0, \beta^0 > 0, \gamma > 0$
2. For k=1, ... until convergence
3. $\mathbf{x}^k = \arg\min_{\mathbf{x}} E_1(\mathbf{x}, \mathbf{v}^{k-1}, \beta_{k-1})$. // inverse problem
4. $\mathbf{v}^k = \arg\min_{\mathbf{v}} E_1(\mathbf{x}^k, \mathbf{v}, \beta_{k-1})$. // regularization
5. $\beta_k = \gamma\beta^{k-1}$
6. End for

This method (which was proposed by [Geman & Yang (2002)]) was notably used by [Zoran & Weiss (2012)] to optimize their EPLL method.

HQS Plug & Play

The fourth step in the HQS method ...

4. $\mathbf{v}^k = \arg\min_{\mathbf{v}} E_1(\mathbf{x}^k, \mathbf{v}) = \arg\min_{\mathbf{v}} \beta_k \|\mathbf{x}^k - \mathbf{v}\|^2 + R(\mathbf{v})$

... can be interpreted as a **denoising** of $\mathbf{x}^k$ with noise variance $\sigma_k^2 = 1/(2\beta^k)$

Replace step 4 by a trained denoiser D_{σ_k}:

1. Initialization: $\mathbf{x}^0, \mathbf{v}^0, \beta^0$
2. For k=1, ... until convergence
3. $\mathbf{x}^k = \arg\min_{\mathbf{x}} E_1(\mathbf{x}, \mathbf{v}^{k-1})$. (inverse problem)
4. $\mathbf{v}^k = D_{\sigma_k}(\mathbf{x}^k)$. (regularization)
5. $\beta_k = \gamma \beta^{k-1}$
6. End for

Obs 1: In this scheme the denoiser needs to be trained for several values of σ_k

Obs 2: The convergence of the P&P HQS method has not been established.

We refer to it here as an historical introduction.

Method of Multipliers

The HQS method requires β to increase during the iterations.

A more proper method which can ensure convergence with a fixed parameter is the *method of multipliers*, which introduces a Lagrange multiplier w_i for each constraint $\mathbf{x}_i = \mathbf{v}_i$:

$$E_2(\mathbf{x}, \mathbf{v}, \mathbf{u}) = F(\mathbf{x}) + \mathbf{u}^T(\mathbf{x} - \mathbf{v}) + \lambda R(\mathbf{v})$$

We can show that

$$\arg\min_{\mathbf{x}} \min_{\mathbf{v}} \max_{\mathbf{u}} E_2(\mathbf{x}, \mathbf{v}, \mathbf{u}) = \arg\min_{\mathbf{x}} E(\mathbf{x})$$

This results in the following algorithm

1. Initialization: $\mathbf{x}^0, \mathbf{v}^0, \mathbf{u}^0, \alpha > 0,$
2. For k=1, ... until convergence
3. $\mathbf{x}^k = \arg\min_{\mathbf{x}} E_2(\mathbf{x}, \mathbf{v}^{k-1}, \mathbf{u}^{k-1})$. // inverse problem
4. $\mathbf{v}^k = \arg\min_{\mathbf{v}} E_2(\mathbf{x}^k, \mathbf{v}, \mathbf{u}^{k-1})$. // regularization
5. $\mathbf{u}^k = \mathbf{u}^{k-1} + \alpha(\mathbf{x}^k - \mathbf{v}^k)$ // gradient ascent on multiplier
6. End for

ADMM - unscaled version

A more flexible version of this algorithm can be constructed, which:

- has milder conditions on D and R
- converges faster

It is obtained by adding the ℓ^2 norm of the constraint to build the augmented Lagrangian:

$$E_3(\mathbf{x}, \mathbf{v}, \mathbf{u}) = F(\mathbf{x}) + \mathbf{u}^T(\mathbf{x} - \mathbf{v}) + \lambda R(\mathbf{v}) + \frac{\rho}{2}\|\mathbf{x} - \mathbf{v}\|^2$$

The corrresponding alternated optimization (where $\alpha = \rho$ is the *unscaled version of ADMM*:

1. Initialization: $\mathbf{x}^0, \mathbf{v}^0, \mathbf{u}^0, \rho > 0$,
2. For k=1, ... until convergence
3. $\mathbf{x}^k = \arg\min_{\mathbf{x}} E_3(\mathbf{x}, \mathbf{v}^{k-1}, \mathbf{u}^{k-1})$. // inverse problem
4. $\mathbf{v}^k = \arg\min_{\mathbf{v}} E_3(\mathbf{x}^k, \mathbf{v}, \mathbf{u}^{k-1})$. // regularization
5. $\mathbf{u}^k = \mathbf{u}^{k-1} + \rho(\mathbf{x}^k - \mathbf{v}^k)$ // gradient ascent on multiplier
6. End for

ADMM - scaled version

ADMM is often written in a more convenient way as follows:
Defining the residual of the constraint as $r = \mathbf{x} - \mathbf{v}$
the "constraint" part of the Lagrangian can be rewritten

$$\mathbf{u}^T r + \frac{\rho}{2}\|r\|^2 = \frac{\rho}{2}\|r + \frac{1}{\rho}\mathbf{u}\|^2 - \frac{1}{2\rho}\|\mathbf{u}\|^2 = \frac{\rho}{2}\|r + \overline{\mathbf{u}}\|^2 - \frac{\rho}{2}\|\overline{\mathbf{u}}\|^2$$

where $\overline{\mathbf{u}} := \frac{1}{\rho}\mathbf{u}$ is the scaled multiplier
With this modification the *scaled ADMM* that results from minimizing

$$E_4(\mathbf{x}, \mathbf{v}, \overline{\mathbf{u}}) = F(\mathbf{x}) + \rho\overline{\mathbf{u}}^T(\mathbf{x} - \mathbf{v}) + \lambda R(\mathbf{v}) + \frac{\rho}{2}\|\mathbf{x} - \mathbf{v}\|^2$$

and the algorithm becomes:

1. Initialization: $\mathbf{x}^0, \mathbf{v}^0, \overline{\mathbf{u}}^0, \rho > 0,$
2. For k=1, ... until convergence
3. $\mathbf{x}^k = \arg\min_{\mathbf{x}} E_4(\mathbf{x}, \mathbf{v}^{k-1}, \overline{\mathbf{u}}^{k-1})$. // inverse problem
4. $\mathbf{v}^k = \arg\min_{\mathbf{v}} E_4(\mathbf{x}^k, \mathbf{v}, \overline{\mathbf{u}}^{k-1})$. // regularization
5. $\overline{\mathbf{u}}^k = \overline{\mathbf{u}}^{k-1} + (\mathbf{x}^k - \mathbf{v}^k)$. // gradient ascent on multiplier
6. End for

ADMM - scaled version

ADMM is often written in a more convenient way as follows:
Defining the residual of the constraint as $r = \mathbf{x} - \mathbf{v}$
the "constraint" part of the Lagrangian can be rewritten

$$\mathbf{u}^T r + \frac{\rho}{2}\|r\|^2 = \frac{\rho}{2}\|r + \frac{1}{\rho}\mathbf{u}\|^2 - \frac{1}{2\rho}\|\mathbf{u}\|^2 = \frac{\rho}{2}\|r + \bar{\mathbf{u}}\|^2 - \frac{\rho}{2}\|\bar{\mathbf{u}}\|^2$$

where $\bar{\mathbf{u}} := \frac{1}{\rho}\mathbf{u}$ is the scaled multiplier
With this modification the *scaled ADMM* that results from minimizing

$$E_4(\mathbf{x}, \mathbf{v}, \bar{\mathbf{u}}) = F(\mathbf{x}) + \lambda R(\mathbf{v}) + \frac{\rho}{2}\|\mathbf{x} - \mathbf{v} + \bar{\mathbf{u}}\|^2 - \frac{\rho}{2}\|\bar{\mathbf{u}}\|^2$$

or expanding the two minimization steps:

1. Initialization: $\mathbf{x}^0, \mathbf{v}^0, \bar{\mathbf{u}}^0, \rho > 0$,
2. For k=1, ... until convergence
3. $\mathbf{x}^k = \arg\min_{\mathbf{x}} F(\mathbf{x}) + \frac{\rho}{2}\|\mathbf{x} - (\mathbf{v}^{k-1} - \bar{\mathbf{u}}^{k-1})\|^2$. // prox on inverse problem
4. $\mathbf{v}^k = \arg\min_{\mathbf{v}} \lambda R(\mathbf{v}) + \frac{\rho}{2}\|(\mathbf{x}^k + \bar{\mathbf{u}}^{k-1}) - \mathbf{v}\|^2$. // prox on regularization
5. $\bar{\mathbf{u}}^k = \bar{\mathbf{u}}^{k-1} + (\mathbf{x}^k - \mathbf{v}^k)$. // gradient ascent on multiplier
6. End for

Plug & Play ADMM

As in the HQS case, step 4 in this algorithm is substituted by a NN-based denoiser that was trained independently of this problem.
Instead of:

4. $\mathbf{v}^k = \arg\min_{\mathbf{v}} \lambda R(\mathbf{v}) + \frac{\rho}{2}\|(\mathbf{x}^k + \overline{\mathbf{u}}^{k-1}) - \mathbf{v}\|^2.$

we write:

4. $\mathbf{v}^k = D_\sigma(\mathbf{x}^k + \overline{\mathbf{u}}^{k-1}).$

where $\sigma = \sqrt{\lambda/\rho}$

Theorem (Plug & Play ADMM [Sreehari et al 2016])

If D_σ is differentiable and its Jacobian J_{D_σ} is **symmetric** *with* **eigenvalues in** $[0,1]$, *plus some mild technical conditions, then:*

- *D_σ is the proximal operator of some energy function R.*
- *Plug & Play ADMM converges to the global infimum of $F(\mathbf{x}) + R(\mathbf{x})$*

Plug & Play ADMM

Theorem (Plug & Play ADMM [Sreehari et al 2016])

If D_σ is differentiable and its Jacobian J_{D_σ} is **symmetric** *with* **eigenvalues in** $[0,1]$, *plus some mild technical conditions, then:*

- *D_σ is the proximal operator of some energy function R.*
- *Plug & Play ADMM converges to the global infimum of $F(\mathbf{x}) + R(\mathbf{x})$*

The proof of this result is based on a result of [Moreau 1965]:

Theorem ([Moreau 1965])

A denoiser D is the proximal operator of an energy R, iff

- *D is non-expansive $\|D(\mathbf{x}) - D(\mathbf{v})\|_2 \leq \|\mathbf{x} - \mathbf{v}\|_2$, and*
- *there exists φ such that $D(\mathbf{x}) \in \partial\varphi(\mathbf{x})$*

Non-expansiveness results from the eigenvalue condition. And the existence of φ from the symmetric Jacobian and Green's theorem.

Plug & Play ADMM (bounded denoisers) [Chan et al 2017]

A slight variation of the P&P-ADMM algorithm allows it to converge under more reasonable conditions.
The modification is similar to the continuation scheme used for HQS:

1. Initialization: $\mathbf{x}^0, \mathbf{v}^0, \overline{\mathbf{u}}^0, \rho^0 > 0, \eta < 1, \gamma > 1$
2. For k=1, ... until convergence
3. $\mathbf{x}^k = \arg\min_{\mathbf{x}} F(\mathbf{x}) + \frac{\rho}{2}\|\mathbf{x} - (\mathbf{v}^{k-1} - \overline{\mathbf{u}}^{k-1})\|^2.$
   // proximal descent on inverse problem
4. $\mathbf{v}^k = D_{\sigma^k}(\mathbf{x}^k + \overline{\mathbf{u}}^{k-1})$ where $\sigma^k = \sqrt{\lambda/\rho^{k-1}}$
5. $\overline{\mathbf{u}}^k = \overline{\mathbf{u}}^{k-1} + (\mathbf{x}^k - \mathbf{v}^k)$. // gradient ascent on multiplier
6. If $\Delta_k \geq \eta \Delta_{k-1}$ then $\rho^k = \gamma \rho^{k-1}$
7. Else $\rho^k = \rho^{k-1}$
8. End if
9. End for

Plug & Play ADMM convergence for bounded denoisers

Theorem ([Chan, Wang & Elgendy 2017])

- *H1: If F has bounded gradients, and*
- *H2: D_σ is bounded (there exists $C > 0$ such that for any image $\mathbf{x} \in \mathbb{R}^N$ $\frac{1}{N}\|D_\sigma(\mathbf{x}) - \mathbf{x}\|^2 \leq \sigma^2 C$)*

Then the iterates $\theta^k = (\mathbf{x}^k, \mathbf{v}^k, \overline{\mathbf{u}}^k)$ of Plug & Play ADMM converge to a fixed point in ℓ^2 norm.

The convergence proof proceeds by showing that $\theta^k = (\mathbf{x}^k, \mathbf{v}^k, \overline{\mathbf{u}}^k)$ is a Cauchy sequence by showing that
$D(\theta^{k+1}, \theta^k) \leq C\delta^k$
for some $C > 0$ and $\delta \in (0, 1)$
This is trivial in case 1 (else case when $\Delta_k < \eta\Delta_{k-1}$).
Otherwise the update rule $\rho_{k+1} = \gamma\rho_k$ and boundedness of the denoiser play an important role:
Indeed $\sigma^k < \gamma^{-1/2}\sigma^{k-1}$, and therefore $\|D_{\sigma^k}(\mathbf{x}^k + \overline{\mathbf{u}}^{k-1}) - (\mathbf{x}^k + \overline{\mathbf{u}}^{k-1}))\|$ also decreases because it is bounded.

Conclusion

- HQS works for EPLL
- RED requires locally homogeneous denoiser with symmetric Jacobian (Wavelet Thresholding, Graph-Laplacian-based denoisers)
- Plug & Play ADMM (fixed ρ) requires non-expansive denoiser (Wavelet Thresholding)
- Plug & Play ADMM (adaptive ρ) requires bounded denoiser (any denoser can be modified to satisfy this condition.

Stephen Boyd.
Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers.
Foundations and Trends® in Machine Learning, 3(1):1–122, 2010.

Antonin Chambolle and Thomas Pock.
An introduction to continuous optimization for imaging.
Acta Numerica, 25:161–319, 2016.

Stanley H. Chan, Xiran Wang, and Omar A. Elgendy.
Plug-and-Play ADMM for Image Restoration: Fixed-Point Convergence and Applications.
IEEE Transactions on Computational Imaging, 3(1):84–98, mar 2017.

J.J. Moreau.
Proximité et dualité dans un espace hilbertien.
Bulletin de la S.M.F., 93(3):273–299, oct 1965.

Tim Meinhardt, Michael Moeller, Caner Hazirbas, and Daniel Cremers.
Learning Proximal Operators: Using Denoising Networks for Regularizing Inverse Imaging Problems.
In *(ICCV) International Conference on Computer Vision*, pages 1781–1790, apr 2017.

Edward T. Reehorst and Philip Schniter.
Regularization by Denoising: Clarifications and New Interpretations.
jun 2018.

Yaniv Romano, Michael Elad, and Peyman Milanfar.
The Little Engine That Could: Regularization by Denoising (RED).
SIAM Journal on Imaging Sciences, 10(4):1804–1844, jan 2017.

Suhas Sreehari, Singanallur V. Venkatakrishnan, Brendt Wohlberg, Gregery T. Buzzard, Lawrence F. Drummy, Jeffrey P. Simmons, and Charles A. Bouman.
Plug-and-Play Priors for Bright Field Electron Tomography and Sparse Interpolation.
IEEE Transactions on Computational Imaging, 2(4):1–1, 2016.