Robust Edge-Preserving Algorithms for PET Image Reconstruction

Jeffrey A. Fessler and Hakan Erdogan

EECS Department
The University of Michigan

Dec. 19, 1997

Outline

- Motivation: PET Image Reconstruction
- Cost Function Description (for a simpler problem)
- Minimization Algorithms
  - Huber Algorithm
  - Optimization Transfer
  - Generalized Huber Algorithm
  - Grouped Coordinate Descent (GCD) Algorithm
- Anecdotal Preliminary Results
- Summary and Future Work

PET Image Reconstruction

Almost all statistical methods for PET image reconstruction are based on the following Poisson statistical model.

$$Y_i \sim \text{Poisson}\{\varepsilon_i s_i \sum_j g_{ij} \lambda_j + r_i\}$$

- Y_i: measured counts in sinogram bins
- λ_j: unknown radiotracer concentration in the jth voxel
- ε_i: ith detector efficiency
- s_i: photon survival probability along ith ray (attenuation)
- g_{ij}: projection matrix
- r_i: random coincidences

Maximum-Likelihood Image Reconstruction

If the Poisson model is valid, it is natural to estimate the emission image λ by finding the "best fit" to the sinogram data, as measured by the log-likelihood:

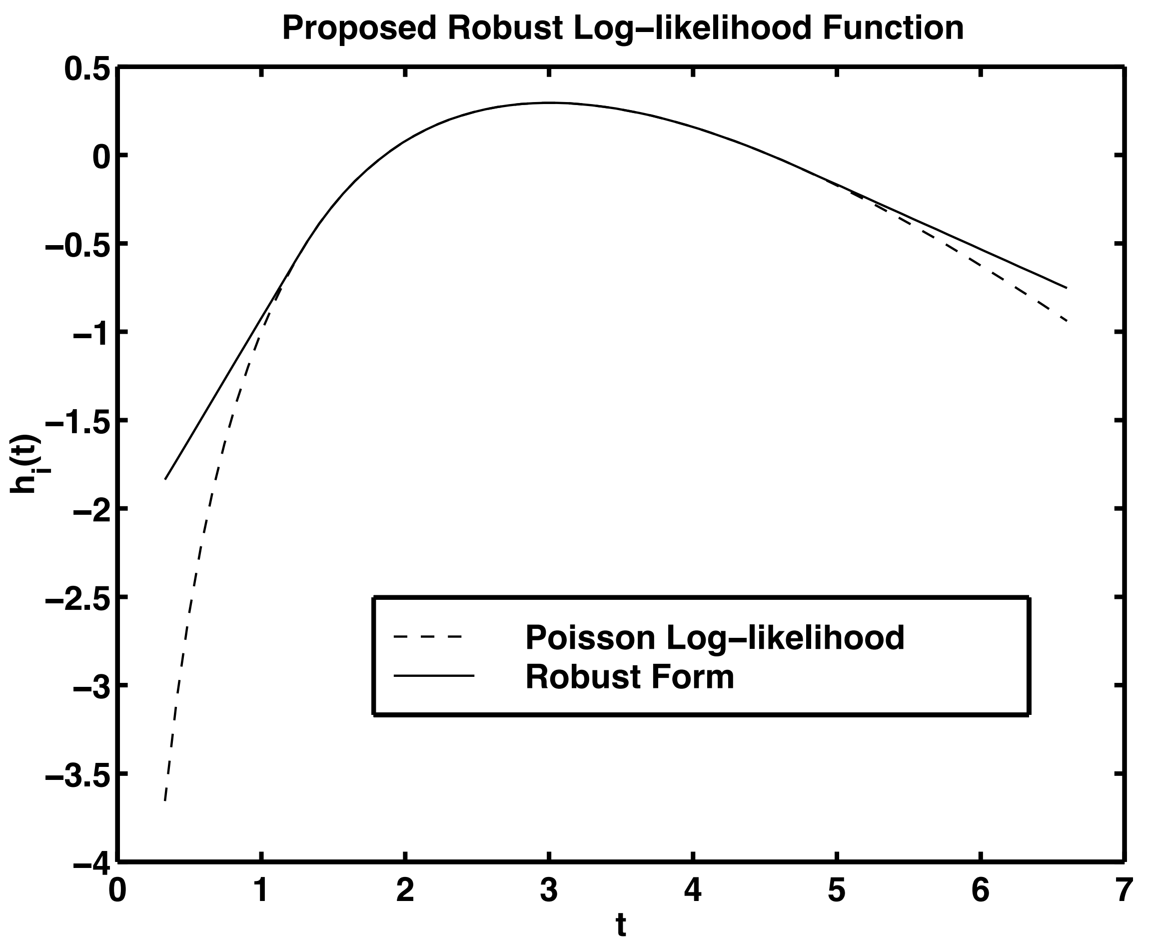
$$\hat{\lambda}_{\mathrm{ML}} \triangleq \arg\max_{\lambda \geq 0} L(\lambda) \quad \text{where} \quad L(\lambda) = \sum_i h_i(\lambda)$$

$$h_i(\lambda) = Y_i \log\left(\varepsilon_i s_i \sum_j g_{ij}\lambda_j + r_i\right) - \left(\varepsilon_i s_i \sum_j g_{ij}\lambda_j + r_i\right).$$

Problem: although the "Poisson" part may be fine, the ε_i's, s_i's, r_i's and g_{ij}'s all contain model errors and random variability.

Especially the s_i's due to noisy transmission scans.

Possible Solution: Robust Log-Likelihood



to be continued...

"Linear" Inverse Problem

$$\boxed{\underline{y} = \boldsymbol{A}\underline{x} + \text{noise}}$$

- $\underline{y}$: noisy measurements (blurred image or sinogram)
- $\underline{x}$: unknown object (true image)
- $\boldsymbol{A}$: known system model
  (each column is a point response function)
- Errors in $\boldsymbol{A}$ partially motivate robust methods

Goal: recover an estimate $\hat{\underline{x}}$ of $\underline{x}$ from $\underline{y}$.

Data-Fit Cost Function

One wants $\hat{\underline{x}}$ to "fit the data," *i.e.* $\underline{y} \approx \boldsymbol{A}\hat{\underline{x}}$ or $\boxed{\underline{y} - \boldsymbol{A}\hat{\underline{x}} \approx \underline{0}}$

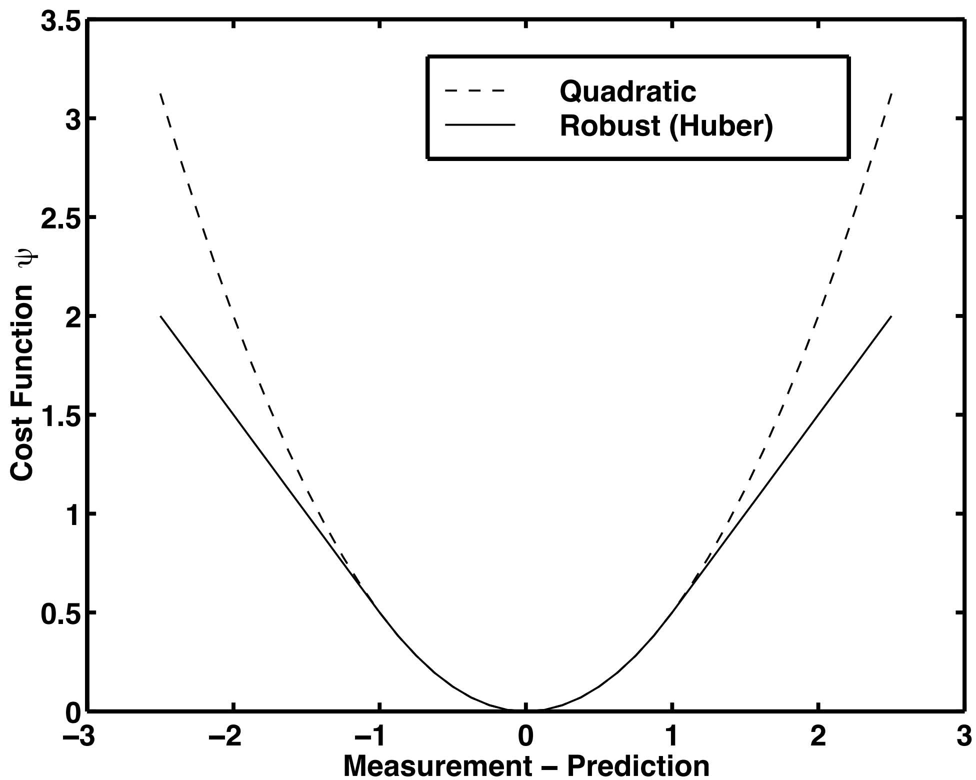
Natural cost function for independent measurement errors:

$$\Phi^{\text{data}}(\underline{x}) = \sum_{i=1}^{m_1} \psi_i^{\text{data}}([\underline{y} - \boldsymbol{A}\underline{x}]_i)$$

- $[\underline{y} - \boldsymbol{A}\underline{x}]_i = y_i - \sum_{j=1}^{p} a_{ij} x_j$
- m_1: length of $\underline{y}$
- ψ_i: convex function.

Traditional choice: $\psi_i(t) = t^2/2$, which is appropriate for Gaussian noise, but is not robust to noise with heavy-tailed distributions.

Robust Data-Fit Cost Function



Huber function: $$\psi(t) = \begin{cases} t^2/2, & |t| \leq \delta, \\ \delta|t| - \delta^2/2, & |t| > \delta \end{cases}$$

1D Robust Estimators

Sample Mean:

$$\hat{\mu} = \arg\min_a \sum_{i=1}^{n} (X_i - a)^2 = \frac{1}{n} X_i$$

Sample Median:

$$\hat{\mu} = \arg\min_a \sum_{i=1}^{n} |X_i - a| = \mathrm{median}\{X_i\}_{i=1}^{n}$$

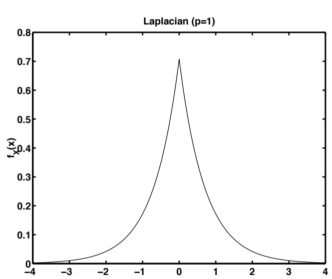
The sample-mean is well known to be very sensitive to outliers.

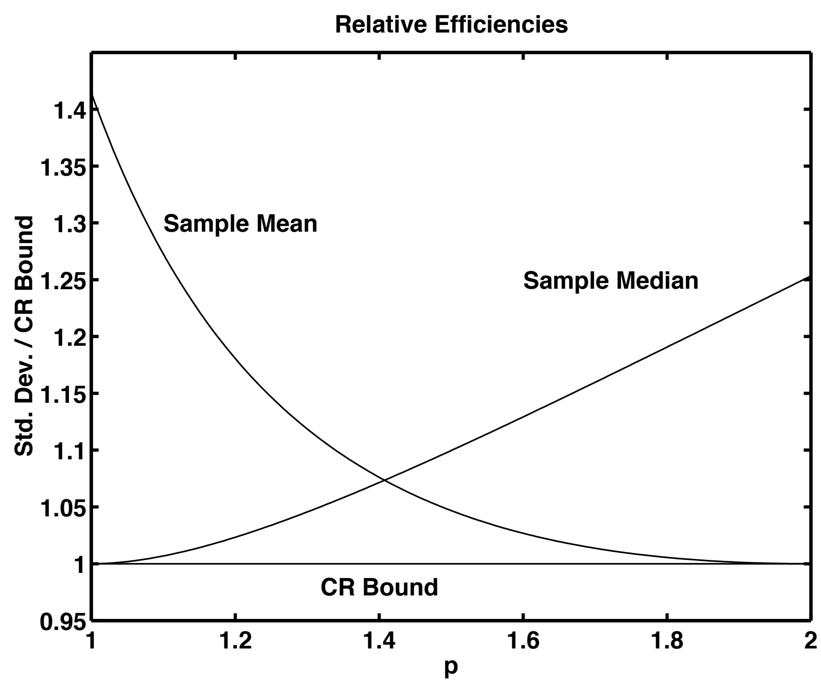
(cf reporting median home prices vs mean home prices)

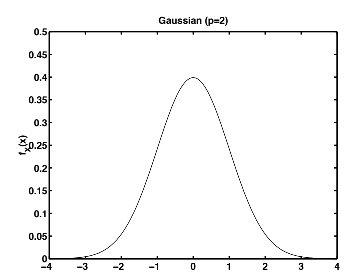
Mean vs Median

Generalized-Gaussian family of pdfs with unit variance:

$$f_X(x;\mu,p) = \frac{p}{2}\frac{1}{\Gamma(1/p)}\sqrt{r_p}e^{-|x-\mu|^p r_p^{p/2}} \quad \text{where } r_p = \frac{\Gamma(3/p)}{\Gamma(1/p)}.$$







Regularization

Minimizing Φ^{data} is inadequate for ill-conditioned inverse problems.

Prior "knowledge" of piece-wise smoothness:

- $x_j - x_{j-1} \approx 0$ (piece-wise constant)
- $x_{j-1} - 2x_j + x_{j+1} \approx 0$ (piece-wise linear)
- $x_j \approx 0$ (support constraints)
- $\ldots$

Combining: $\boldsymbol{C}\underline{x} \approx \underline{z}$

Regularized cost function: $\boxed{\Phi(\underline{x}) = \Phi^{\text{data}}(\underline{x}) + \Phi^{\text{penalty}}(\underline{x}),}$

$$\Phi^{\text{penalty}}(\underline{x}) = \sum_{i=1}^{m_2} \psi_i^{\text{penalty}}([\boldsymbol{C}\underline{x} - \underline{z}]_i)$$

Example: Roughness Penalty (Gibbs Prior)

$$\boldsymbol{D}_n = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 \\ & & \ddots & \ddots & \\ 0 & 0 & 0 & 1 & -1 \end{bmatrix} \qquad \boldsymbol{C} = \begin{bmatrix} \boldsymbol{I}_{n_y} \otimes \boldsymbol{D}_{n_x} \\ \boldsymbol{D}_{n_y} \otimes \boldsymbol{I}_{n_x} \end{bmatrix}$$

where $\otimes$ denotes the Kronecker matrix product.

If $\underline{z} = \underline{0}$ and $\mathcal{N}_j$ is the four pixel neighborhood of pixel j, then

$$\boxed{\Phi^{\text{penalty}}(\underline{x}) = \sum_j \sum_{k \in \mathcal{N}_j} \psi_{j,k}(x_j - x_k)}$$

Conventional (Tikhonov-Miller) regularization: $\psi(t) = t^2/2$.

For edge-preserving image recovery, need non-quadratic $\psi(\cdot)$.

Unified Cost Function

$$\boxed{\Phi(\underline{x}) = \sum_{i=1}^{m} \psi_i([\boldsymbol{B}\underline{x} - \underline{c}]_i)}$$

Regularized edge-preserving cost function is a special case:

$$\Phi(\underline{x}) = \Phi^{\text{data}}(\underline{x}) + \Phi^{\text{penalty}}(\underline{x}), \quad \boldsymbol{B} = \begin{bmatrix} \boldsymbol{A} \\ \boldsymbol{C} \end{bmatrix}, \quad \underline{c} = \begin{bmatrix} \underline{y} \\ \underline{z} \end{bmatrix}$$

Optimization problem:

$$\hat{\underline{x}} = \arg\min_{\underline{x}} \Phi(\underline{x}) \quad \text{or} \quad \boxed{\hat{\underline{x}} = \arg\min_{\underline{x} \geq \underline{0}} \Phi(\underline{x}).}$$

Optimization

Simple in quadratic case where $\psi_i(t) = t^2/2 \ \forall i$

$$\underline{\hat{x}} = \arg\min_{\underline{x}} \frac{1}{2} \|\boldsymbol{B}\underline{x} - \underline{c}\|^2 = (\boldsymbol{B}'\boldsymbol{B})^{-1}\boldsymbol{B}'\underline{c}$$

Good (fast converging) iterative algorithms:

- Preconditioned conjugate gradients
- Coordinate descent (Gauss-Siedel)

Challenging for non-quadratic ψ_i's

Very challenging for non-convex ψ_i's

Proposition: algorithms tailored to structure of Φ can outperform general purpose optimization methods.

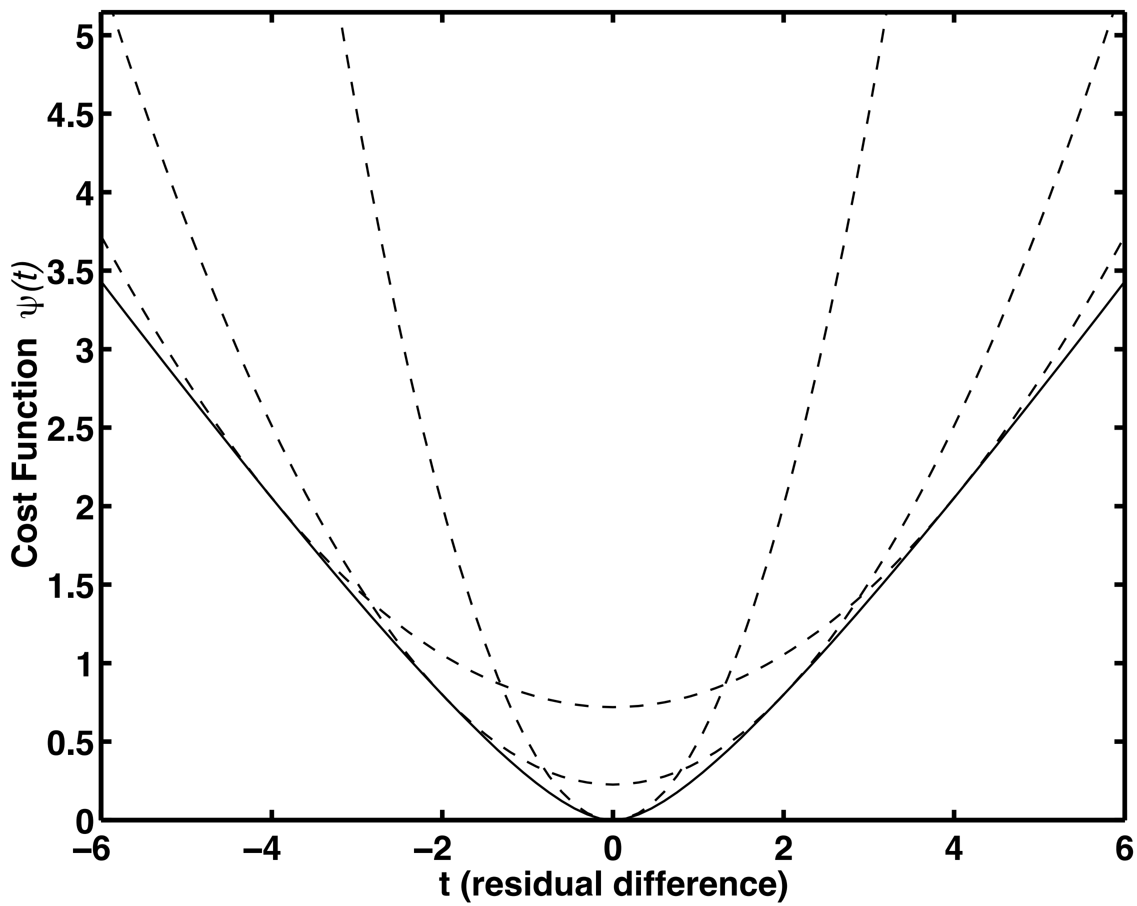
But cannot solve all...

Assumptions

- $\boldsymbol{B}$ has full column rank, so $\boldsymbol{M} > \boldsymbol{0} \;\Rightarrow\; \boldsymbol{B}'\boldsymbol{M}\boldsymbol{B} > \boldsymbol{0}$
  (Easily achieved with sensible regularization design)
- Each individual cost-function satisfies:
  - ψ is symmetric
  - ψ is everywhere differentiable (and therefore continuous)
  - $\dot{\psi}(t) = d/dt\,\psi(t)$ is non-decreasing (and hence ψ is convex)
  - $\boxed{\omega_\psi(t) = \dot{\psi}(t)/t}$ is non-increasing for $t \geq 0$
  - $\omega_\psi(0) = \lim_{t\to 0} \dot{\psi}(t)/t$ is finite and nonzero *i.e.* $0 < \omega_\psi(0) < \infty$
- Φ has a unique minimizer
  (Easily ensured with perturbation of regularizer)

(rules out entropy, $|t|^p$) to understand ω, look at...

Tangent Parabolas



$\omega_\psi(t_0)$ is the curvature of the parabola that is tangent at t_0

Unconstrained Solution

$$\boxed{\Phi(\underline{x}) = \sum_{i=1}^{m} \psi_i([\boldsymbol{B}\underline{x} - \underline{c}]_i)}$$

Column gradient:

$$\nabla\Phi(\underline{x}) = \boldsymbol{B}'\boldsymbol{\Omega}(\underline{x})(\boldsymbol{B}\underline{x} - \underline{c}), \qquad \nabla\Phi(\underline{x})\big|_{\underline{x}=\hat{\underline{x}}} = \boldsymbol{0}$$

$$\text{where} \quad \boldsymbol{\Omega}(\underline{x}) = \operatorname{diag}\{\omega_{\psi_i}([\boldsymbol{B}\underline{x} - \underline{c}]_i)\}$$

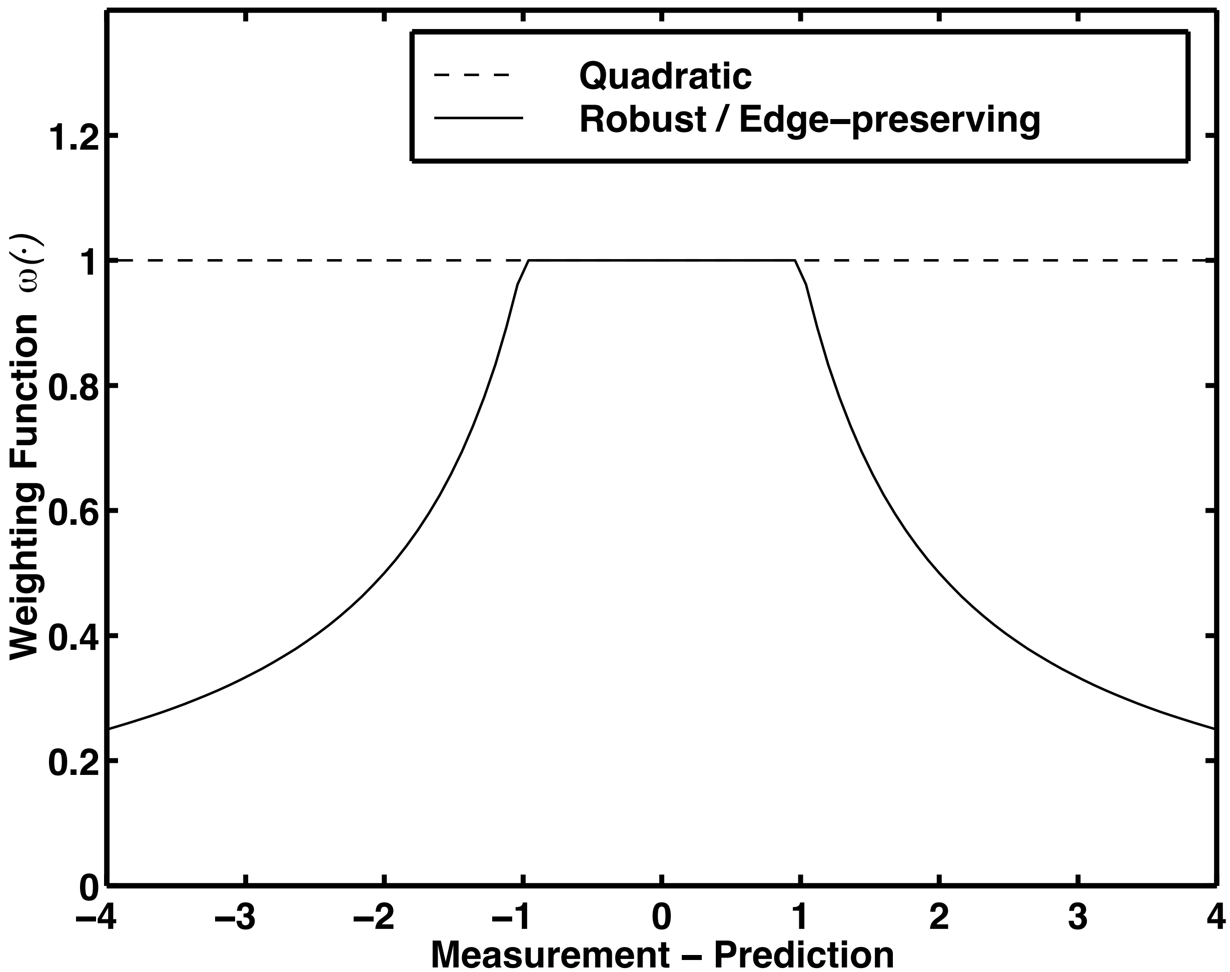
Unconstrained solution:

$$\begin{aligned} \hat{\underline{x}} &= [\boldsymbol{B}'\boldsymbol{\Omega}(\hat{\underline{x}})\boldsymbol{B}]^{-1}\boldsymbol{B}'\boldsymbol{\Omega}(\hat{\underline{x}})\underline{c} \\ &= \arg\min_{\underline{x}} \frac{1}{2}(\underline{c} - \boldsymbol{B}\underline{x})'\boldsymbol{\Omega}(\hat{\underline{x}})(\underline{c} - \boldsymbol{B}\underline{x}) \end{aligned}$$

(ala WLS, but <u>weights</u> depend on estimate $\hat{\underline{x}}$, hence nonlinear)

Therefore need iterative algorithm...

Weighting Functions ω_ψ



Newton-Raphson Algorithm

$$\boxed{\underline{x}^{n+1} = \underline{x}^n - [\boldsymbol{B}'\boldsymbol{\Lambda}(\underline{x}^n)\boldsymbol{B}]^{-1}\nabla\Phi(\underline{x}^n)}$$

where

$$\boldsymbol{\Lambda}(\underline{x}^n) = \operatorname{diag}\left\{\ddot{\psi}_i([\boldsymbol{B}\underline{x} - \underline{c}]_i)\right\}$$

Advantage:

- Super-linear convergence rate (if convergent)

Disadvantages:

- Requires twice-differentiable ψ_i's
- Not guaranteed to converge
- Not guaranteed to monotonically decrease Φ
- Does not enforce nonnegativity constraint
- Impractical for image recovery due to matrix inverse

Generic remedy: bound-constrained Quasi-Newton algorithms

Huber Algorithm (1981)

Recall $\qquad \underline{\hat{x}} = [\boldsymbol{B}'\boldsymbol{\Omega}(\underline{\hat{x}})\boldsymbol{B}]^{-1}\boldsymbol{B}'\boldsymbol{\Omega}(\underline{\hat{x}})\underline{c} = \underline{\hat{x}} - [\boldsymbol{B}'\boldsymbol{\Omega}(\underline{\hat{x}})\boldsymbol{B}]^{-1}\nabla\Phi(\underline{\hat{x}})$

Successive Substitutions:

$$\boxed{\underline{x}^{n+1} = \underline{x}^{n} - [\boldsymbol{B}'\boldsymbol{\Omega}(\underline{x}^{n})\boldsymbol{B}]^{-1}\nabla\Phi(\underline{x}^{n})}$$

Advantages:

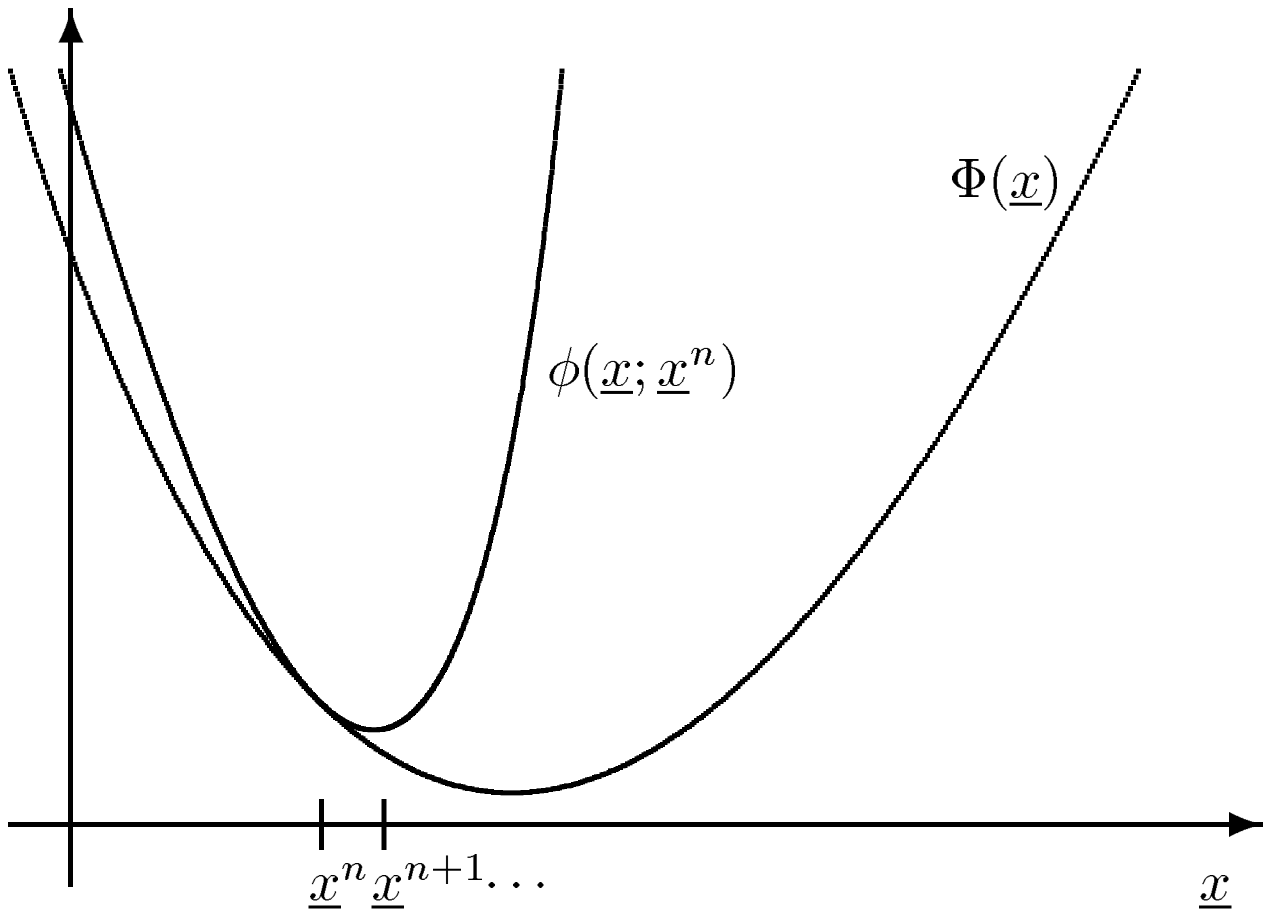
- Monotonically decreases Φ
- Converges globally to unique minimizer (not shown by Huber)

Disadvantages:

- Does not enforce nonnegativity constraint
- Impractical for image recovery due to matrix inverse

Successive substitutions is often not convergent. Why here?

Optimization Transfer



Monotone Decrease in Φ

Minimizing surrogate function ϕ ensures a monotone decrease in the original cost function Φ if:

- $\phi(\underline{x}^n; \underline{x}^n) = \Phi(\underline{x}^n)$
- $\nabla_{\underline{x}} \phi(\underline{x}; \underline{x}^n)\Big|_{\underline{x}=\underline{x}^n} = \nabla \Phi(\underline{x})\big|_{\underline{x}=\underline{x}^n}$
- $\Phi(\underline{x}) \leq \phi(\underline{x}; \underline{x}^n)$

Huber's Algorithm:

$$\underline{x}^{n+1} = \arg\min_{\underline{x}} \phi^{\text{Huber}}(\underline{x}; \underline{x}^n)$$

$$\phi^{\text{Huber}}(\underline{x}; \underline{x}^n) = \frac{1}{2}(\underline{c} - \boldsymbol{B}\underline{x})' \boldsymbol{\Omega}(\underline{x}^n)(\underline{c} - \boldsymbol{B}\underline{x}) + k(\underline{x}^n)$$

- The above 3 (sufficient) conditions are satisfied by ϕ^{Huber} !
- ϕ^{Huber} is quadratic form in $\underline{x}$, so the surrogate is a paraboloid

Optimization Transfer in 2D

Generalized Huber Algorithm

$$\boxed{\underline{x}^{n+1} = \underline{x}^n - \boldsymbol{M}_n^{-1} \nabla \Phi(\underline{x}^n)}$$

where

$$\boldsymbol{M}_n \geq \boldsymbol{B}' \boldsymbol{\Omega}(\underline{x}^n) \boldsymbol{B}$$

Advantages:

- Monotonically decreases Φ
- Converges globally to unique minimizer
- Can choose $\boldsymbol{M}_n$ to be easily invertible, e.g. diagonal. (Or splitting matrices more generally)

Disadvantages:

- Does not enforce nonnegativity constraint in general
- Converges slower than Huber algorithm

Separable Paraboloid

One can use the convexity of the Huber surrogate function to define a second surrogate function that is **separable**:

$$\phi^{\text{Huber}}(\underline{x};\underline{x}^n) \leq \phi^{SP}(\underline{x};\underline{x}^n) \triangleq \sum_j q_j(x_j - x_j^n;\underline{x}^n)$$

where

$$q_j(t;\underline{x}^n) = \sum_{i=1}^{m} \alpha_{ij}\omega_i([\boldsymbol{B}\underline{x}^n - \underline{c}]_i)\frac{1}{2}\left(\frac{b_{ij}}{\alpha_{ij}}t + \boldsymbol{B}\underline{x}^n - \underline{c}\right)^2,$$

$$\alpha_{ij} = \frac{|b_{ij}|}{\sum_{j=1}^{p} |b_{ik}|}.$$

Minimizing the separable paraboloid ϕ^{SP} is trivial, especially compared to minimizing a paraboloid.

Separable Paraboloid Algorithm

$$\underline{x}^{n+1} = \left[\underline{x}^n - \frac{\dot{q}_j(0;\underline{x}^n)}{\ddot{q}_j(0;\underline{x}^n)}\right]_+ = \left[\underline{x}^n - \operatorname{diag}\left\{\frac{1}{\ddot{q}_j(0;\underline{x}^n)}\right\} \nabla\Phi(\underline{x}^n)\right]_+$$

Advantages:

- Monotonically decreases Φ
- Converges globally to unique minimizer
- No matrix inversion required
- Can enforce nonnegativity constraint
- Parallelizable (all pixels updated simultaneously)

Disadvantages:

- Very slow convergence (ala EM algorithm)
  Solution: update only a subset of the pixels simultaneously

Grouped Coordinate Descent Algorithm

Construct a separable paraboloidal surrogate function but for only a (large) **subset** of the pixels.

Pixel Groups (2x3)

1	5	3	1	5	3	1	5
4	2	6	4	2	6	4	2
1	5	3	1	5	3	1	5
4	2	6	4	2	6	4	2
1	5	3	1	5	3	1	5
4	2	6	4	2	6	4	2

Pixels separated => decoupled => fast convergence

Many pixels per subiteration => parallelizable

Grouped Coordinate Descent Algorithm

Advantages:

- Monotonically decreases Φ
- Converges globally to unique minimizer
- No matrix inversion required
- Can enforce nonnegativity constraint
- Parallelizable (all pixels updated simultaneously)
- **Fast convergence**

Disadvantages:

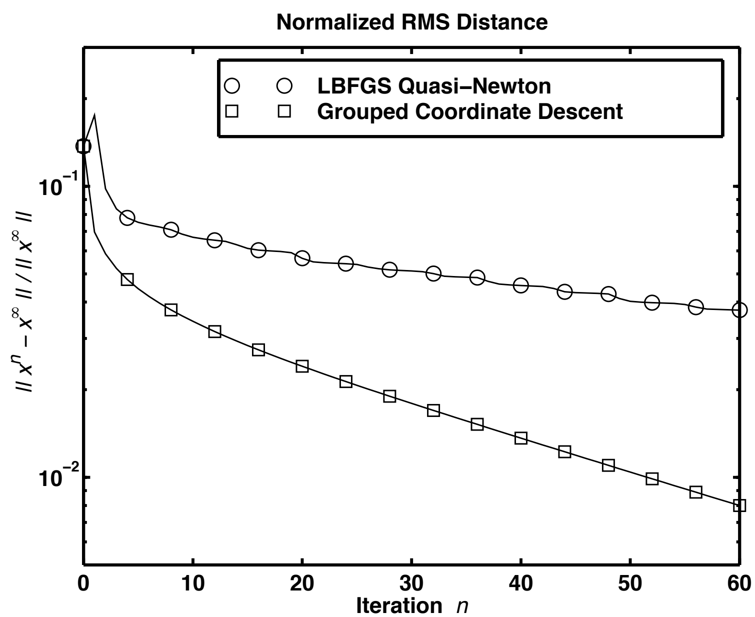
- Slightly less parallelizable.
- Slightly more complicated implementation
- More complicated to explain...
- Difficult to exploit structure of $\boldsymbol{B}$
  (e.g. FFTs for shift-invariant PSF, separable blur in PET)

PET Transmission Example

- 12 minute transmission scan from ECAT EXACT (single slice)
- 0.921M prompt coincidences
- 160 radial by 192 angular samples
- 128×128 attenuation map with 4.5 mm pixel size

Normalized RMS Distance

$$\frac{\|\underline{x}^n - \underline{x}^\infty\|}{\|\underline{x}^\infty\|}$$ where $\underline{x}^\infty$: 400 iterations of single-coordinate descent



LBFGS: Limited Memory Bound Constrained Quasi-Newton Method

(R. Byrd, P. Lu, J. Nocedal, R. Schnabel, C. Zhu)

(Thanks to Web Stayman for interfacing LBFGS with ASPIRE.)

Summary

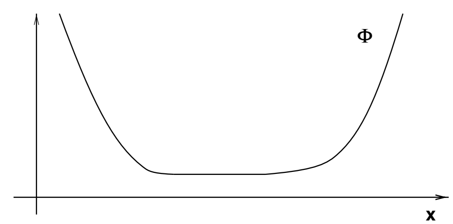
Grouped Coordinate Descent Algorithm

- Accommodates non-quadratic cost function (for noise robustness and preserving edges)
- Monotonically decreases Φ
- Converges globally to unique minimizer
- Easily accommodates nonnegativity constraint
- Parallelizable
- Converges faster than a general-purpose optimization method

---

Future Work:

- Convergence proof for multiple minima:
- PET emission reconstruction problem
- ...



Convergence

From R. Meyer "Sufficient conditions for the convergence of monotonic mathematical programming algorithms," J. Comput. System. Sci., 1976.

Let $\mathcal{M}$ be a point to set mapping such that on G $\mathcal{M}$ is uniformly compact, upper semi-continuous, and strictly monotonic with respect to the function Φ.

If $\{\underline{x}^n\}$ is any sequence generated by the algorithm corresponding to $\mathcal{M}$, then

- all accumulation points of $\{\underline{x}^n\}$ will be fixed points,
- $\Phi(\underline{x}^n) \to \Phi(\underline{x}^\star)$ where $\underline{x}^\star$ is a fixed point,
- $\|\underline{x}^{n+1} - \underline{x}^n\| \to 0$, and
- either $\{\underline{x}^n\}$ converges or the accumulation points of $\{\underline{x}^n\}$ form a continuum.