CSE 337: Introduction to Medical Imaging

Lecture 10: Iterative Reconstruction Methods

Klaus Mueller

Computer Science Department

Stony Brook University

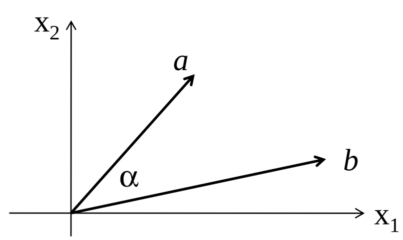
Introduction

Iterative methods are advantageous in these cases:

- limited number of projections
- irregularly-spaced and -angled projections
- non-straight ray paths (example: refraction in ultrasound imaging)
- corrective measures during reconstruction (example: metal artifacts)
- presence of statistical (Poisson) noise and scatter (mainly in functional imaging: SPECT, PET)

Foundations: Vectors

Consider two vectors, *a* and *b*



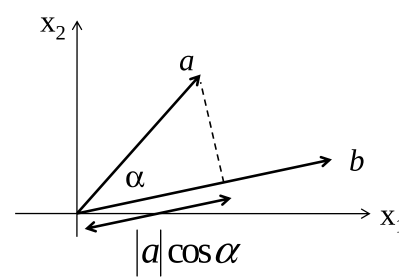
$$a = \vec{a} = [a_1 \ a_2], \qquad |a| = \sqrt{a_1^2 + a_2^2}$$

$$b = \vec{b} = [b_1 \ b_2], \qquad |b| = \sqrt{b_1^2 + b_2^2}$$

Foundations: Scalar Projection

Scalar projection of *a* onto *b*:

$$|a| \cos\alpha = a \cdot \frac{b}{|b|}$$



The *dot product*:

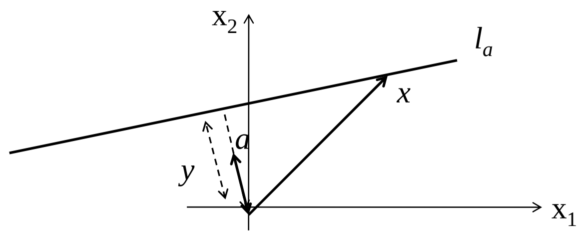
$$a \cdot b = \vec{a} \cdot \vec{b}^T = [a_1 \ a_2] \cdot [b_1 \ b_2]^T = a_1 b_1 + a_2 b_2$$
$$= |a| \cdot |b| \cos\alpha$$

→ the scalar projection is the dot product with |*b*| =1 (unit vector)

$$|b| = \sqrt{b_1^2 + b_2^2} = 1$$

Foundations: Line Equation

$$a_1x_1 + a_2x_2 = y$$
$$|a| = \sqrt{a_1^2 + a_2^2} = 1$$



The vector *a* is the unit vector normal to the line l_a

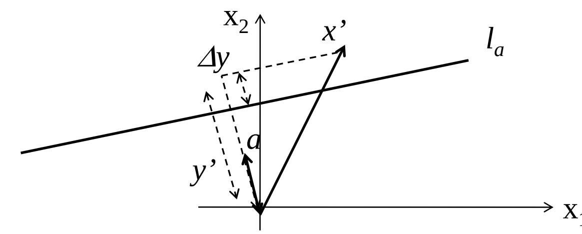
The length *y* is the perpendicular distance of l_a to the origin

For any point *x*:

- if *x* is on l_a then the scalar projection of *x* onto *a* will be:

$$x \cdot a = y$$

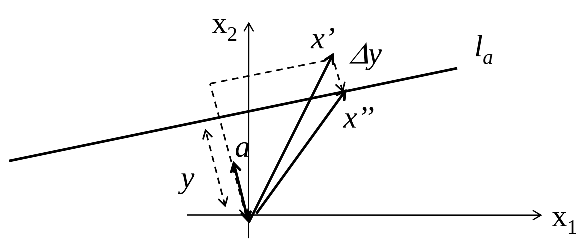
Foundations: Distance From Line



For any other point *x'* not on l_a the scalar projection of *x'* onto *a* will be:

$$x \cdot a = y' = y + \Delta y$$

Foundations: Closest Point



The closest point to *x'* on l_a is *x''*, computed by:

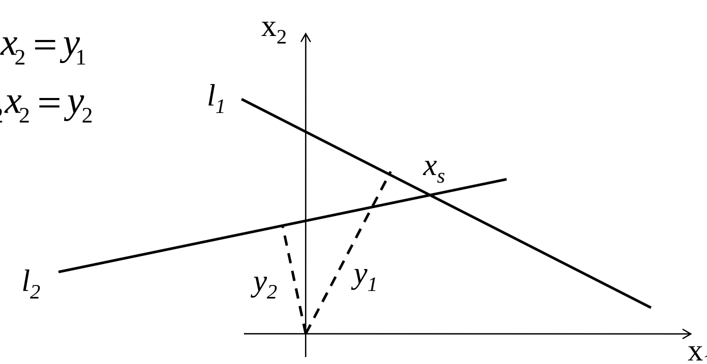
$$\begin{aligned} x'' &= x' - dy \\ &= x' - (x' \cdot a - y) \\ &= x' + (y - x' \cdot a) \end{aligned}$$

Foundations: Solving an Equation System

Assume you have two equations to solve for solution point $x_s=(x_1,x_2)$

- the intersection of the two lines

$$a_{11}x_1 + a_{12}x_2 = y_1$$
$$a_{21}x_1 + a_{22}x_2 = y_2$$

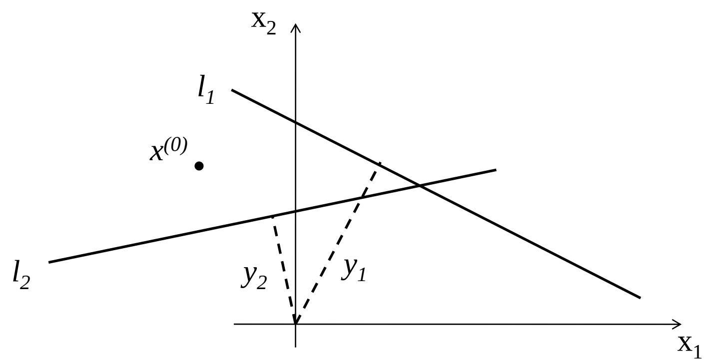


Foundations: Iterating to Solution

Of course, you could solve this equation via Gaussian elimination

- we shall take an iterative approach instead

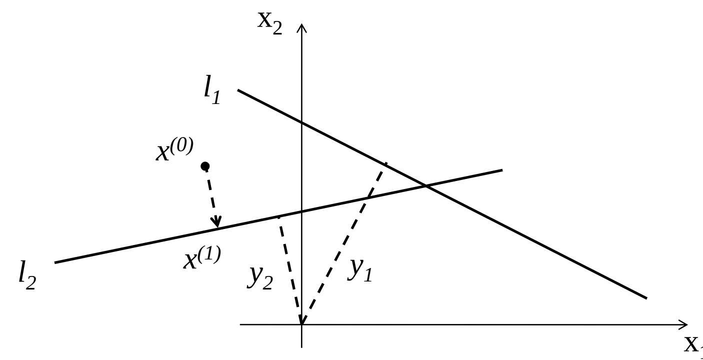
Start with some point $x^{(0)}=(x_1,x_2)$



Foundations: Iterating to Solution

Pick an equation (line, say l_2) and find the closest point to $x^{(0)}$

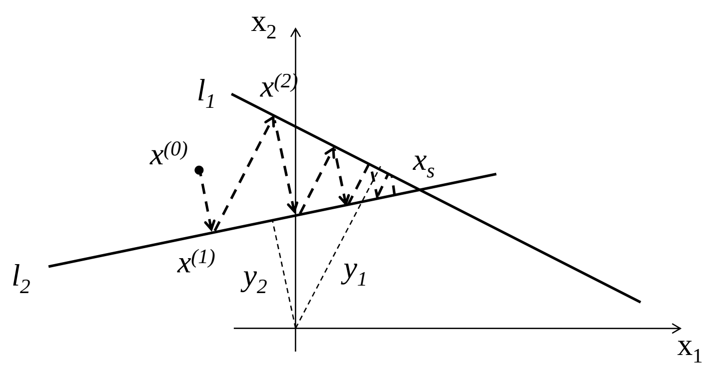
- use the approach outlined before
- this gives a new point $x^{(1)}$



Foundations: Iterating to Solution

Iteratively

- pick alternate equations (lines) and project
- the solution will *converge* towards x_s
- the more iterations the closer the convergence

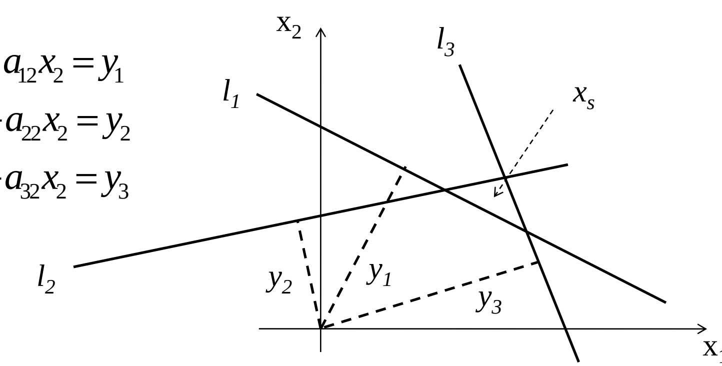


Foundations: Inconsistent Equations

Real life data:

- typically equations (the data) are not consistent
- you may have more equations (data) than unknowns or not enough
- solution falls within a *convex* shape spanned by the intersection set
- need further criteria to determine the true solution (some *prior model*)

$$a_{11}x_1 + a_{12}x_2 = y_1$$
$$a_{21}x_1 + a_{22}x_2 = y_2$$
$$a_{31}x_1 + a_{32}x_2 = y_3$$



Foundations: Determining the True Solution

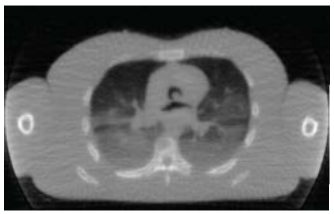
Need further criteria to determine the true solution

Use some *prior model*

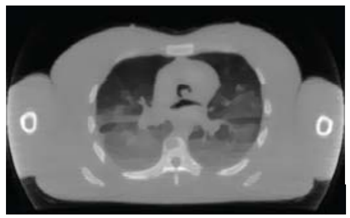
- smoothness, approximate shape, sharp edges, …
- incorporate this model into the reconstruction procedure

Example:

- enforce smoothness by intermittent blurring
- but at the same time preserve edges



streak artifacts, good edges



smooth, good edges

Foundations: Extension to Higher Dimensions

Three dimensions:

- 3 equations with 3 unknowns

N dimensions:

- *N* equations with *M* unknowns
- *M* can be less or greater than *N*
- inconsistent (most often) or not

Specifics to Medical Imaging

In medical imaging:

- *M* unknown voxels (depending on desired object resolution)
- *N* known measurements (pixels in the projection images)
- represent voxels and pixels as vectors *V* and *P*, respectively

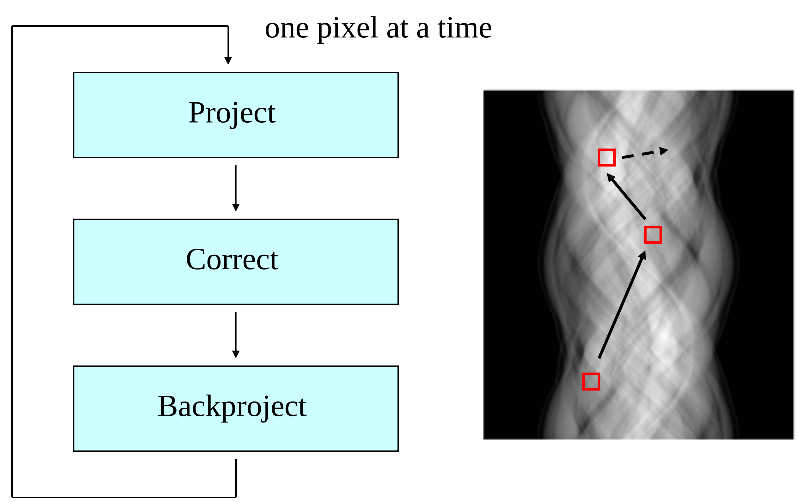
$$w_{11}v_1 + w_{12}v_2 + \ldots w_{1M}v_M = p_1$$
$$w_{21}v_1 + w_{22}v_2 + \ldots w_{2M}v_M = p_2$$
$$\ldots.$$
$$w_{N1}v_1 + w_{N2}v_2 + \ldots w_{NM}v_M = p_N$$

- this gives rise to a system $W \cdot V = P$

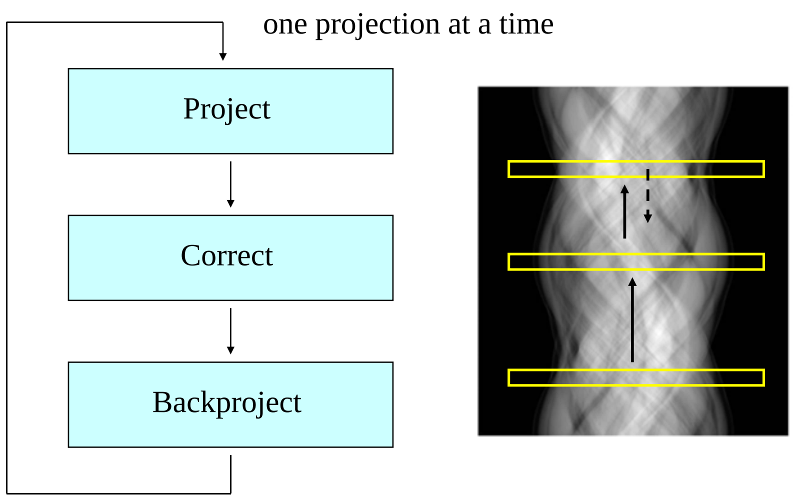
Iterate either by

- ray by ray (Algebraic Reconstruction Technique, ART)
- image by image (Simultaneous ART, SART)
- all data at once (SIRT)

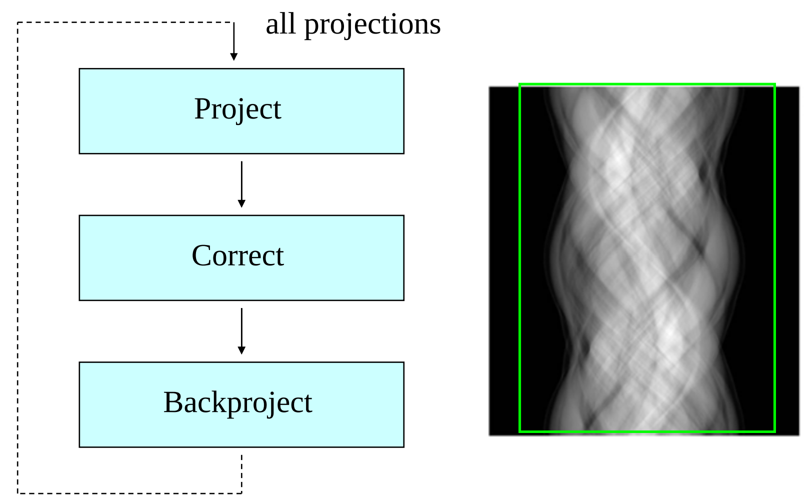
Iterative Update Schedule: ART



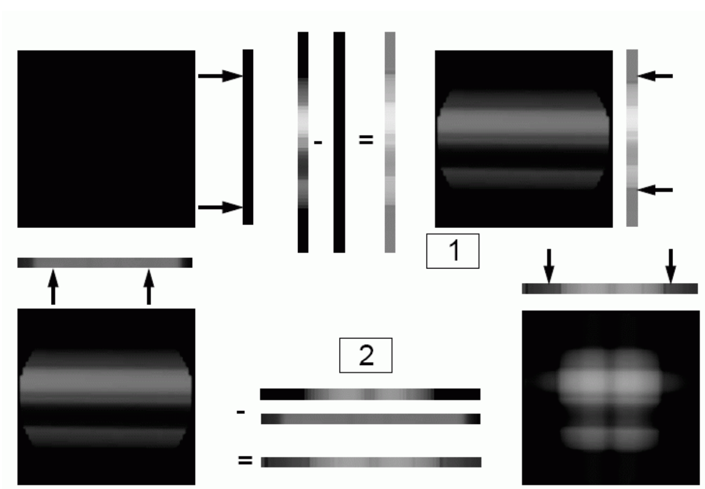
Iterative Update Schedule: SART



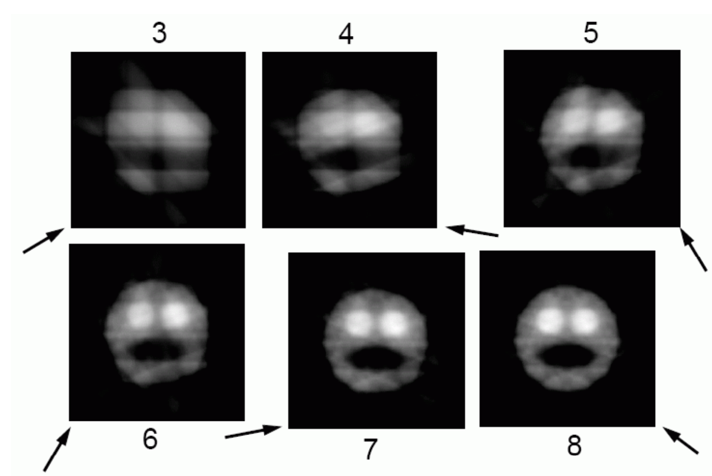
Iterative Update Schedule: SIRT



Iterative Reconstruction Demonstration: SART



Iterative Reconstruction Demonstration: SART



SART

Iteratively solves $W \cdot V = P$

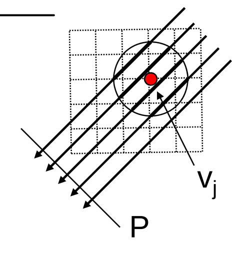
$$v_j^{k+1} = v_j^k + \lambda \frac{\sum_i \frac{p_i - \sum_j v_j^k w_{ij}}{\sum_j w_{ij}} w_{ij}}{\sum_i w_{ij}}$$

SART

Projection

Projection (into pixel)

$$v_j^{k+1} = v_j^k + \lambda \frac{\sum_i \frac{p_i - \boxed{\sum_l v_l^k w_{il}}}{\boxed{\sum_l w_{il}}} w_{ij}}{\sum_i w_{ij}}$$



SART

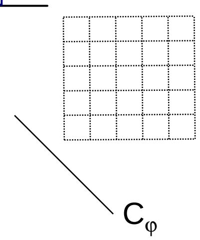
Correction factor computation

Projection (into pixel)

Scanned pixel

Normalization at pixel *i*

$$v_j^{k+1} = v_j^k + \lambda \frac{\sum_i \boxed{\frac{p_i - \boxed{\sum_l v_l^k w_{il}}}{\boxed{\sum_l w_{il}}}} w_{ij}}{\sum_i w_{ij}}$$



SART

Backprojection

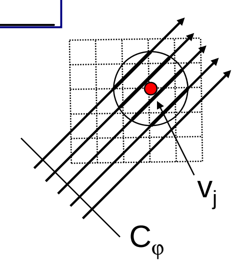
Projection (into pixel)

Backprojection (into voxel)

Scanned pixel

Normalization at pixel *i*

$$v_j^{k+1} = v_j^k + \lambda \frac{\boxed{\sum_i \frac{p_i - \boxed{\sum_l v_l^k w_{il}}}{\boxed{\sum_l w_{il}}} w_{ij}}}{\boxed{\sum_i w_{ij}}}$$



SART

Voxel normalization

Projection (into pixel)

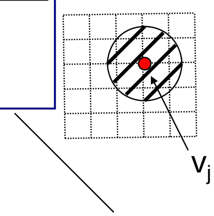
Backprojection (into voxel)

Scanned pixel

Normalization at pixel *i*

Normalization at voxel *j*

$$v_j^{k+1} = v_j^k + \lambda \frac{\sum_i \dfrac{p_i - \sum_l v_l^k w_{il}}{\sum_l w_{il}} w_{ij}}{\sum_i w_{ij}}$$



SART

Voxel update

Projection (into pixel)

Backprojection (into voxel)

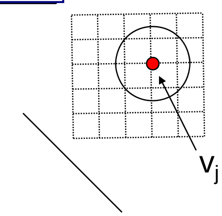
Scanned pixel

Normalization at pixel *i*

New (*k*+1) and previous (*k*) values of voxel *j*

Normalization at voxel *j*

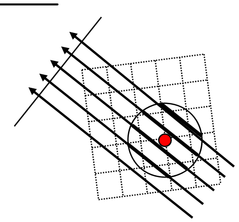
$$v_j^{k+1} = v_j^k + \lambda \frac{\sum_i \dfrac{p_i - \sum_l v_l^k w_{il}}{\sum_l w_{il}} w_{ij}}{\sum_i w_{ij}}$$



SART

Next projection

$$v_j^{k+1} = v_j^k + \lambda \frac{\sum_i \dfrac{p_i - \sum_l v_l^k w_{il}}{\sum_l w_{il}} w_{ij}}{\sum_i w_{ij}}$$



Statistical Techniques

Algebraic methods do not model statistical effects in the underlying data

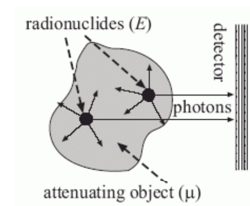
- this is OK for CT (within reason)

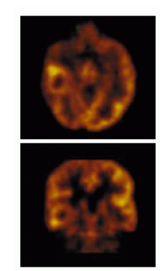
However, the emission of radiation from radionuclides is highly statistical

- the direction is chosen at random
- similar metabolic activities may not emit the same radiation
- not all radiation is actually collected (collimators reject many photons)
- in low-dose CT, noise is also a significant problem

Need a reconstruction method that can accounts for these statistical effects

- Maximum Likelihood – Expectation Maximization (ML-EM) is one such method





Foundations: The Poisson Distribution

Also called the *law of rare events*

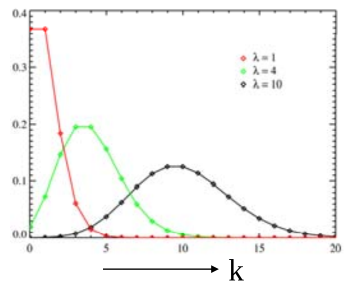
- it is the binomial distribution of *k* as the number of trials *n* goes to infinity

$$\lim_{n \to \infty} \Pr(X = k) = \lim_{n \to \infty} \binom{n}{k} p^k (1-p)^{n-k}$$

- with $p = \lambda / n$

$$f(k;\lambda) = \frac{e^{-\lambda}\lambda^k}{k!}$$

λ: expected number of events (the mean) in a given time interval

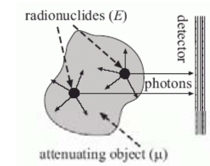


Some examples for Poisson-distributed events:

- the number of phone calls at a call center per minute
- the number of spelling errors a secretary makes while typing a single page
- the number of soldiers killed by horse-kicks each year in each corps in the Prussian cavalry
- the number of positron emissions in a radio nucleotide in PET and SPECT
- the number of annihilation events in PET and SPECT

Overall Concept of ML-EM

There are three types of variables



#1: The observed data y(d):

- the detector readings

#2: The unobserved (latent) data x(b):

- the photon emission activities in the pixels (the tissue), x(b)
- these give rise to the detector readings
- they follow a Poisson distribution

#3: The model parameters λ(b):

- these cause the emissions
- they are the metabolic activities (state) of interest
- the emissions only approximate those

→ they represent the expectations (means, λ) of the resulting Poisson distribution causing the readings at the detectors

Overall Concept of ML-EM

There is a many-to-one mapping of parameters → data

Since there is a many-to-one mapping, many objects are probable to have produced the observed data

- the object reconstruction (the *image*) having the highest such probability is the *maximum likelihood estimate* of the original object

Goal:

- estimate the model parameters using the observed data

Solution:

- EM will converge to a solution of maximum likelihood (but not necessarily the global maximum)

Overall Concept of ML-EM

Initialization step: choose an initial setting of the model parameters

Then proceed to EM, which has two steps, executed iteratively:

- E (expectation) step: estimate the unobserved data from the current estimate of the model parameters and the observed data
- M (maximization) step: compute the maximum-likelihood estimate of the model parameters using the estimated unobserved data

Stop when converged

Initialize model parameters *p*

↓

E-Step: estimate unobserved data *x* using *p* and observed data *y*

↓

M-Step: compute ML-estimate of *p* using *x*

↓

return if converged

Maximum Likelihood Expectation Maximization (ML-EM)

After combining the E-step and the ML-step:

$$v_j^{k+1} = \frac{v_j^k}{\sum_i w_{ij}} \sum_i \frac{p_i}{\sum_j v_j^k w_{ij}}$$

Maximum Likelihood Expectation Maximization (ML-EM)

Maximizes the likelihood of the values of (object) voxels *j*, given values at (detector) pixels *i*

New (*k*+1) and previous (*k*) values of voxel *j*

Backprojection (into voxel *j*)

$$v_j^{k+1} = \frac{v_j^k}{\boxed{\sum_i w_{ij}}} \boxed{\sum_i \frac{p_i}{\boxed{\sum_j v_j^k w_{ij}}}}$$

Normalization at voxel *j*

Projection (into pixel *i*)

Algorithm Comparison

SART:

- projection ordering important
- ensure that consecutively selected projections are approximately orthogonal
- random selection works well in practice

EM:

- convergence slow if all projections are applied before voxel update
- use OS-EM (Ordered Subsets EM): only a subset of projections are applied per iteration