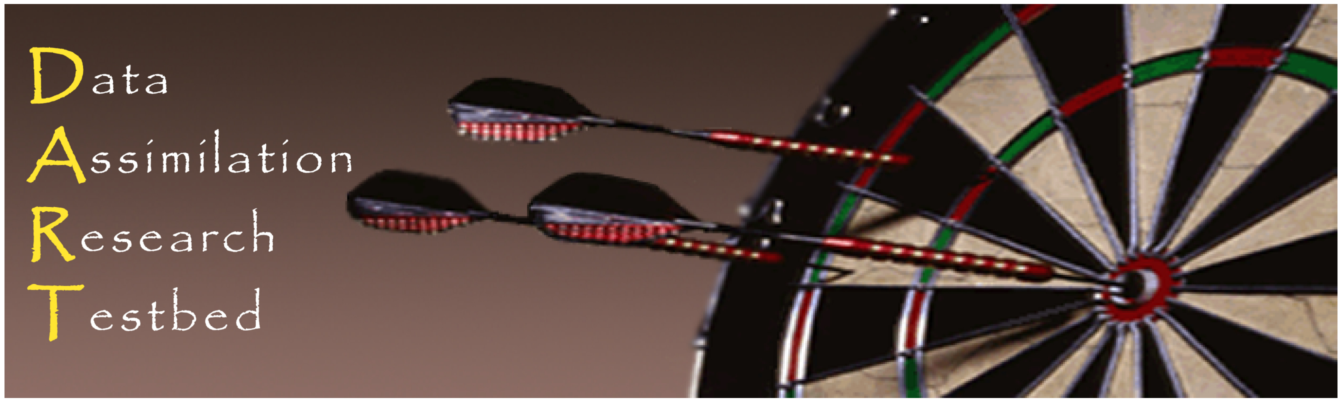
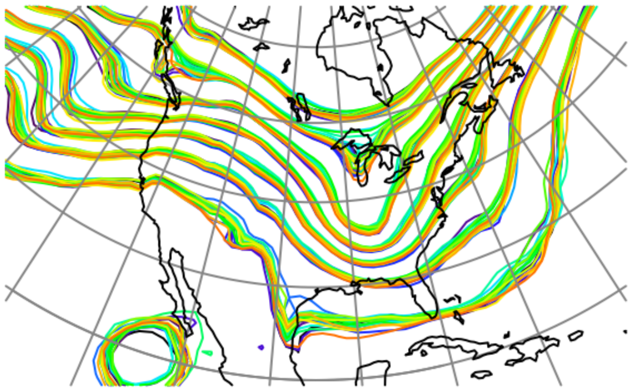


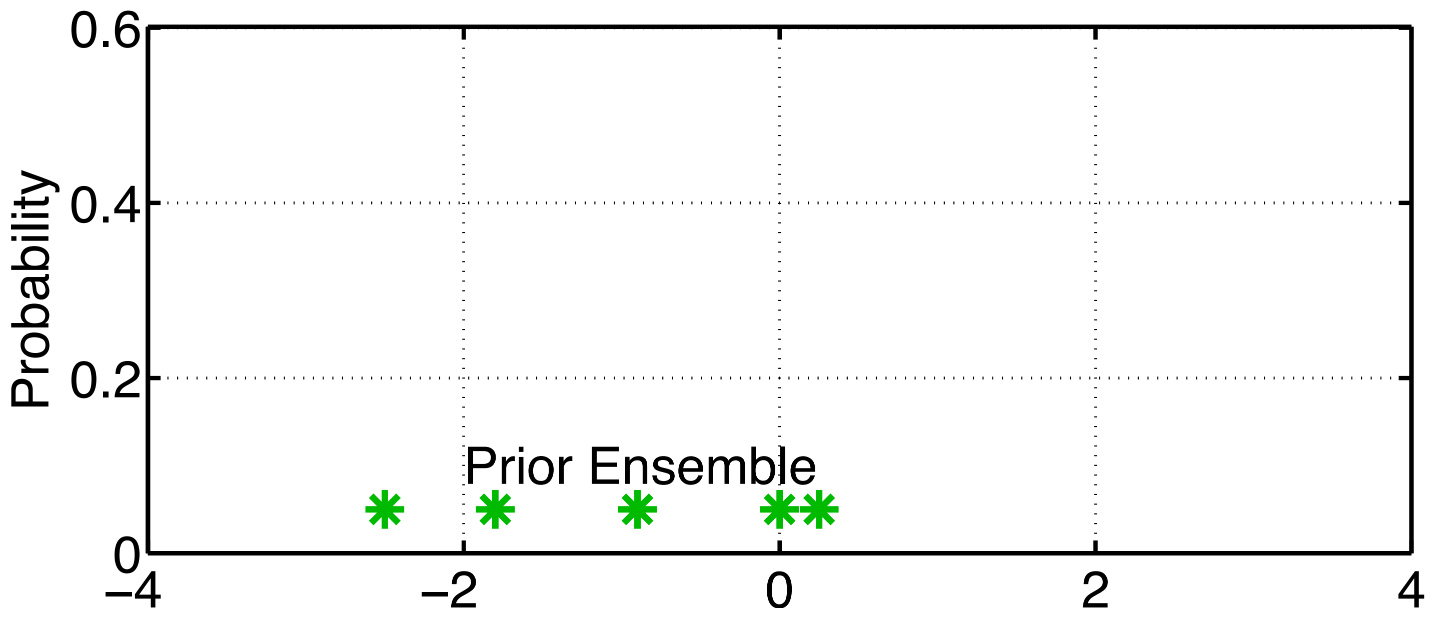
DART Tutorial Section 6: Other Updates for an Observed Variable



Product of Two Gaussians

$$p(A|BC) = \frac{p(B|AC)p(A|C)}{p(B|C)} = \frac{p(B|AC)p(A|C)}{\int p(B|x)p(x|C)dx}$$

Ensemble filters: <u>Prior is available as finite sample.</u>

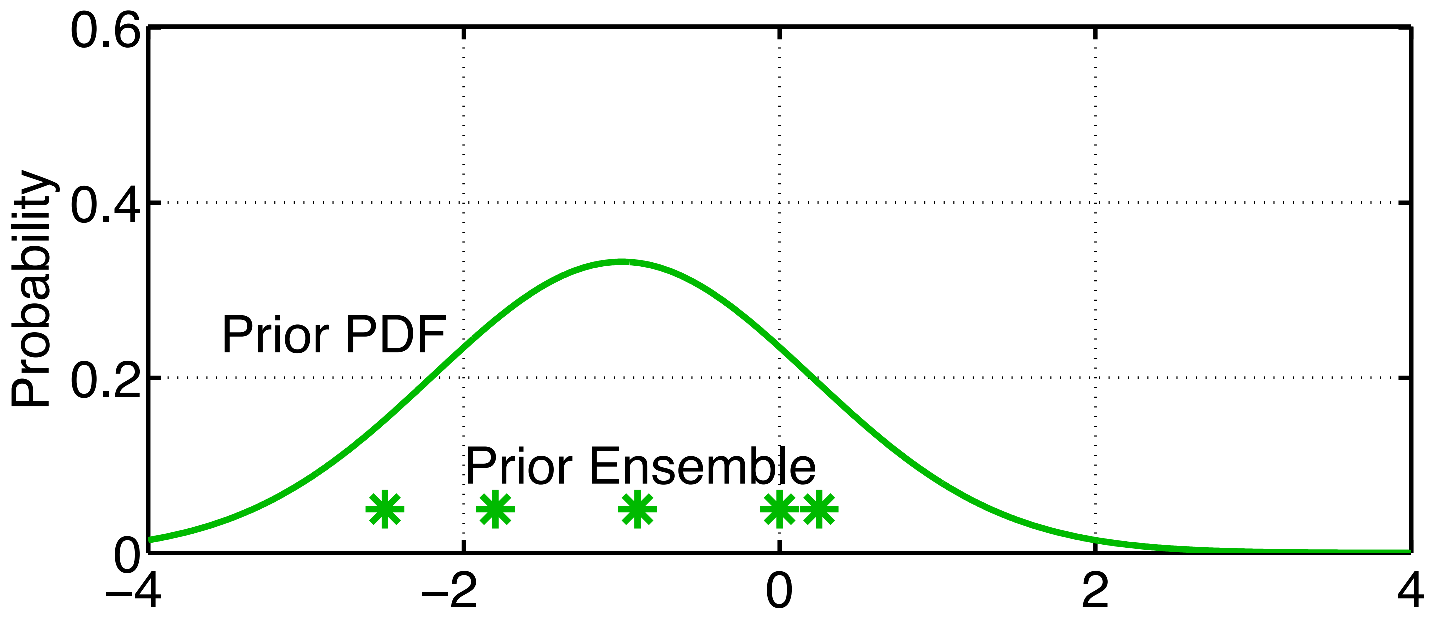


Don’t know much about properties of this sample.
May naively assume it is random draw from ‘truth’.

Product of Two Gaussians

$$p(A|BC) = \frac{p(B|AC)p(A|C)}{p(B|C)} = \frac{p(B|AC)p(A|C)}{\int p(B|x)p(x|C)dx}$$

How can we take product of sample with continuous likelihood?

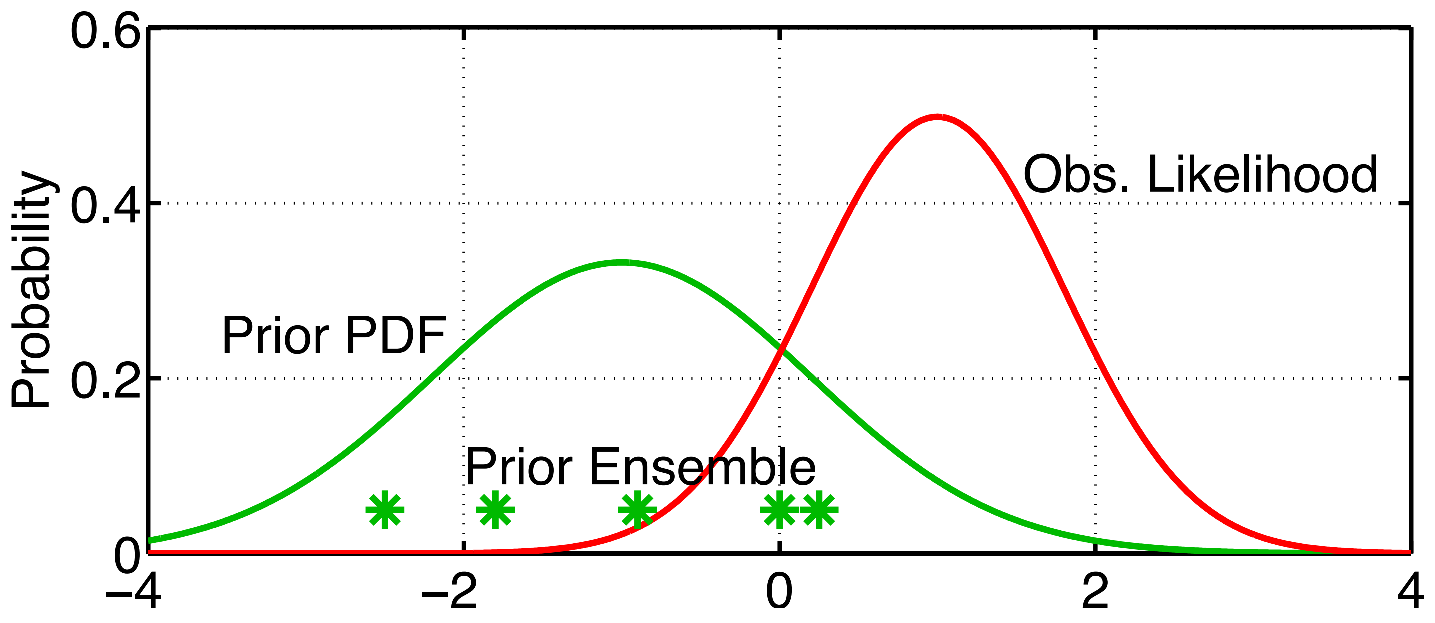


Fit a continuous (Gaussian for now) distribution to sample.

Product of Two Gaussians

$$p(A|BC) = \frac{p(B|AC)p(A|C)}{p(B|C)} = \frac{p(B|AC)p(A|C)}{\int p(B|x)p(x|C)dx}$$

Observation likelihood usually continuous (nearly always Gaussian).

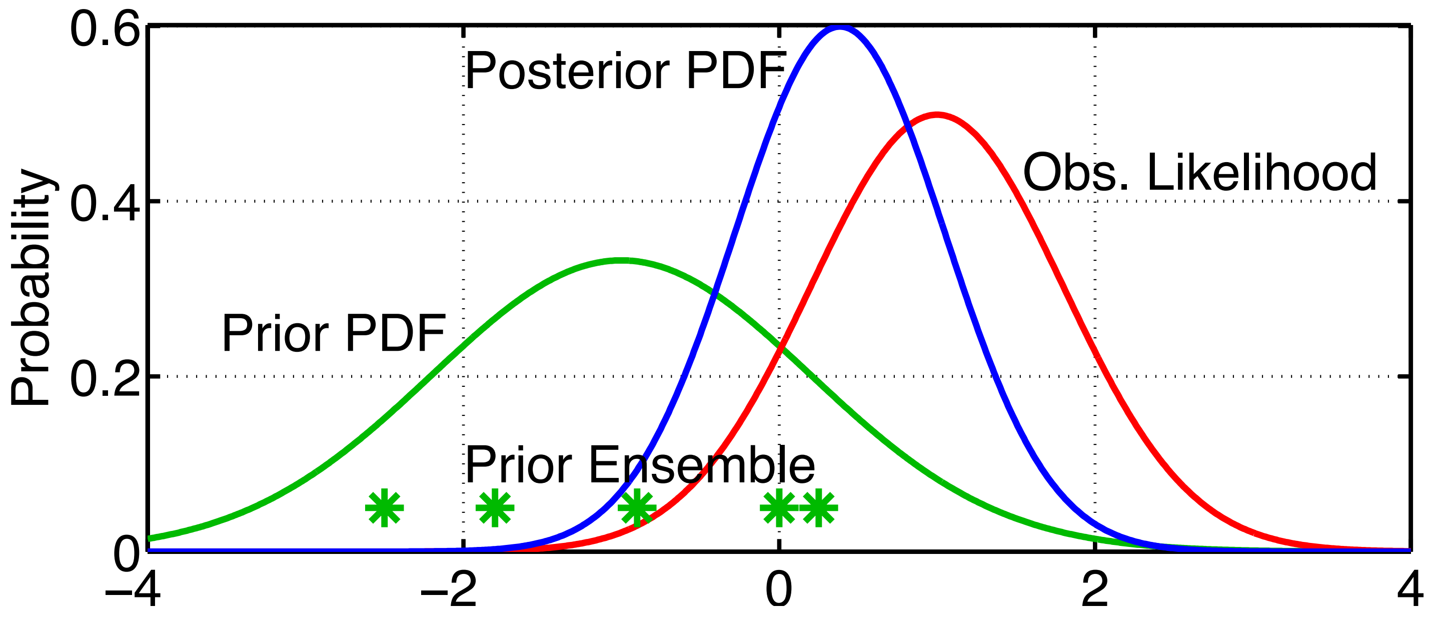


If Obs. Likelihood isn't Gaussian, can generalize methods below.
For instance, can fit set of Gaussian kernels to obs. likelihood.

Product of Two Gaussians

$$p(A|BC) = \frac{p(B|AC)p(A|C)}{p(B|C)} = \frac{p(B|AC)p(A|C)}{\int p(B|x)p(x|C)dx}$$

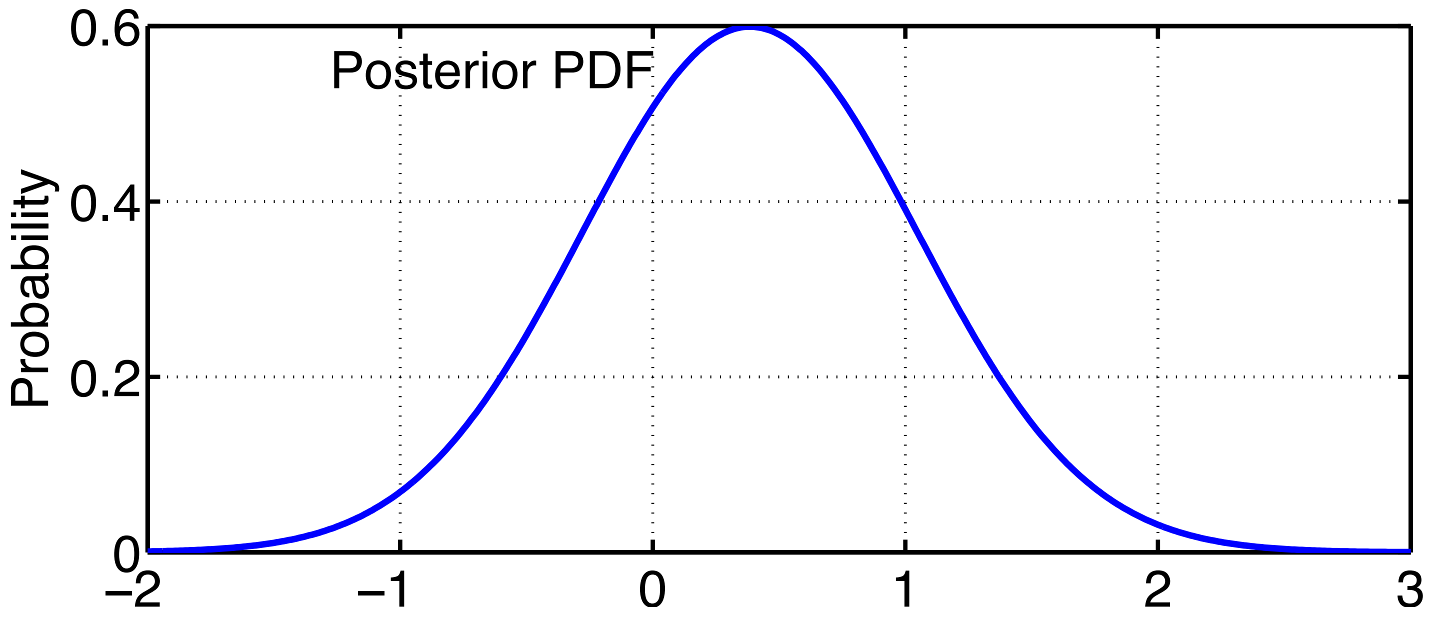
Product of prior Gaussian fit and Obs. likelihood is Gaussian.



Computing continuous posterior is simple.
BUT, need to have a SAMPLE of this PDF.

Sampling Posterior PDF:

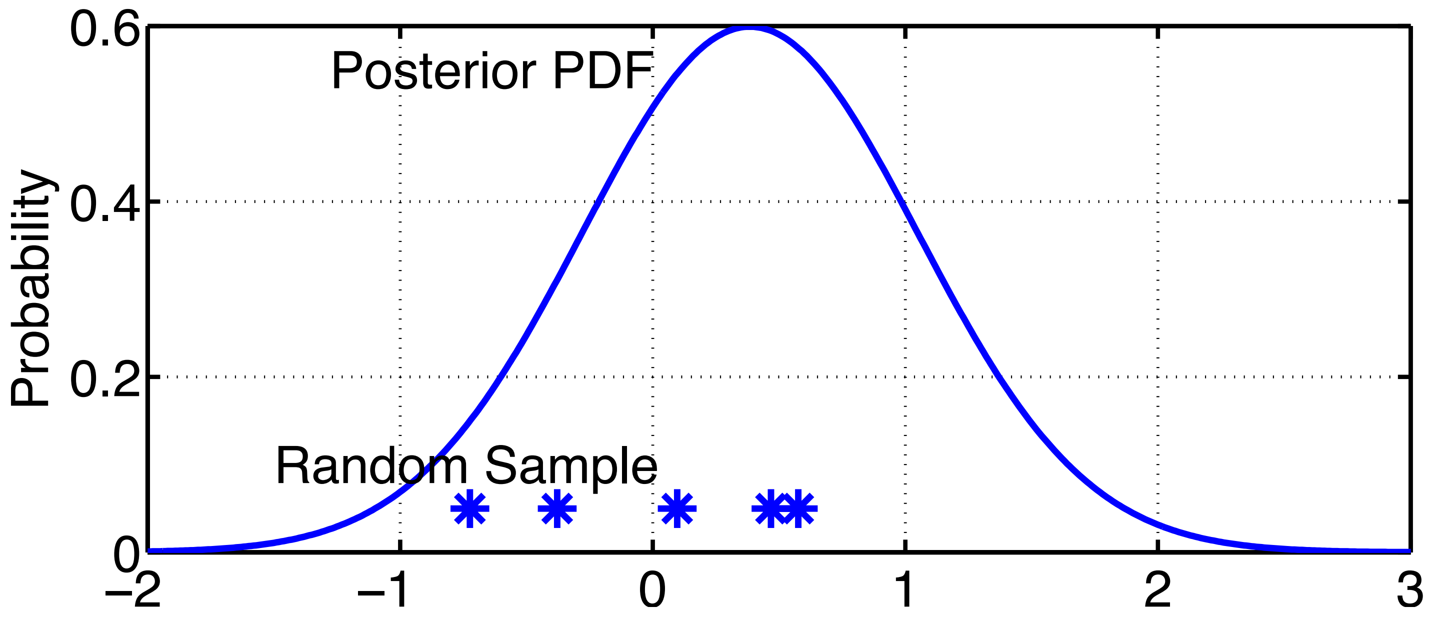
There are many ways to do this.



Exact properties of different methods may be unclear.
Trial and error still best way to see how they perform.
Will interact with properties of prediction models, etc.

Sampling Posterior PDF:

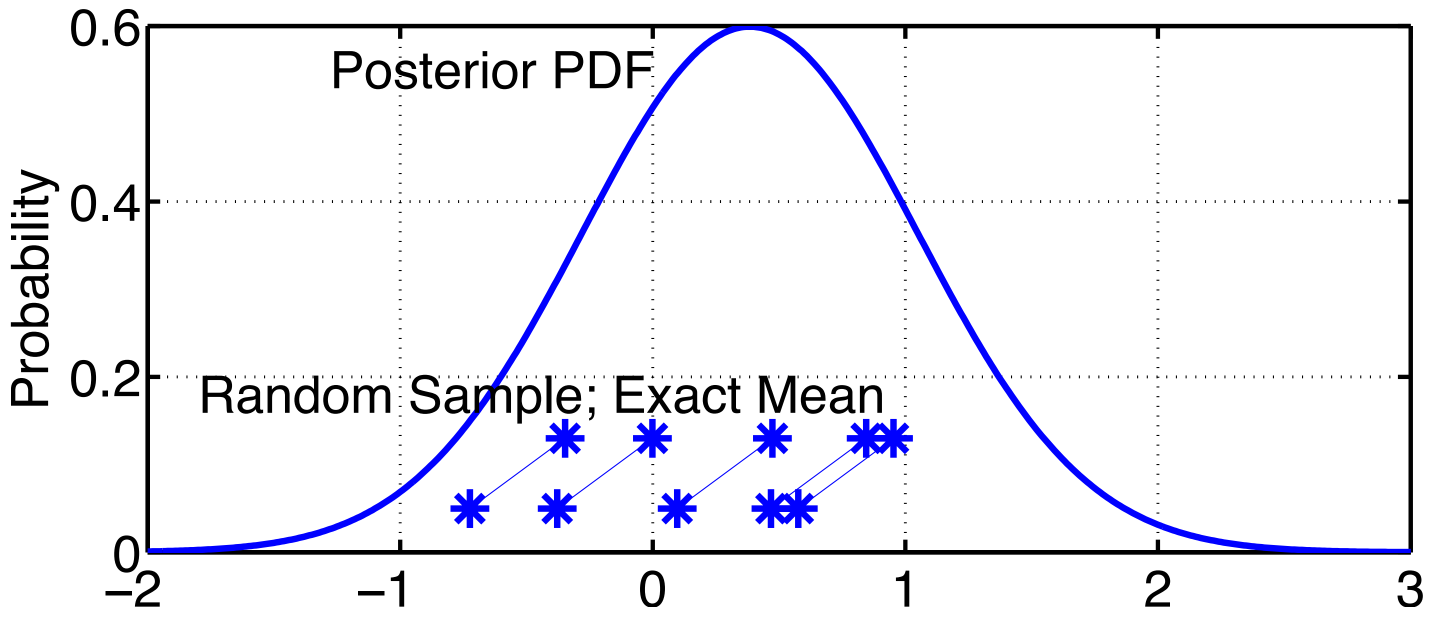
Just draw a random sample (*filter_kind=5* in *assim_tools_nml*).



NOTE: When trying filter_kinds other than 1, *sort_obs_inc* in *assim_tools_nml* should be *.true.* (see section 10).

Sampling Posterior PDF:

Just draw a random sample (*filter_kind=5* in *assim_tools_nml*).

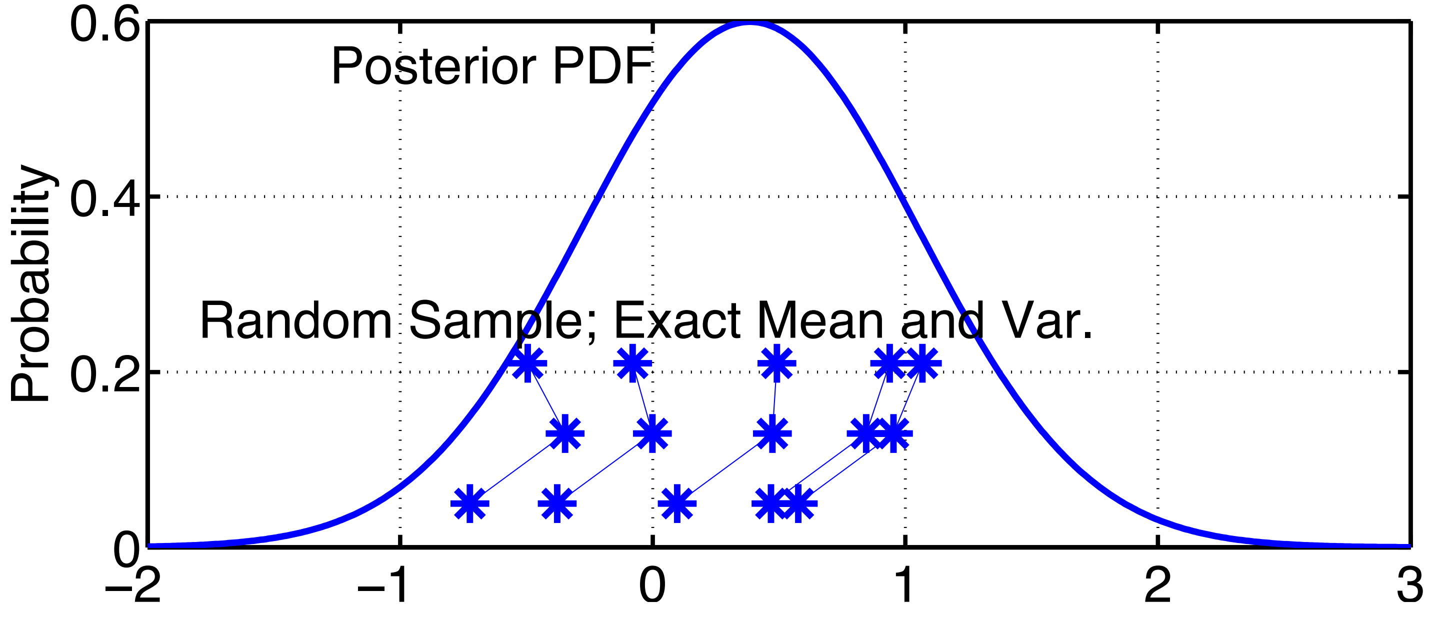


Can ‘play games’ with this sample to improve (modify) its properties.

Example: Adjust the mean of the sample to be exact.

Sampling Posterior PDF:

Just draw a random sample (*filter_kind=5* in *assim_tools_nml*).

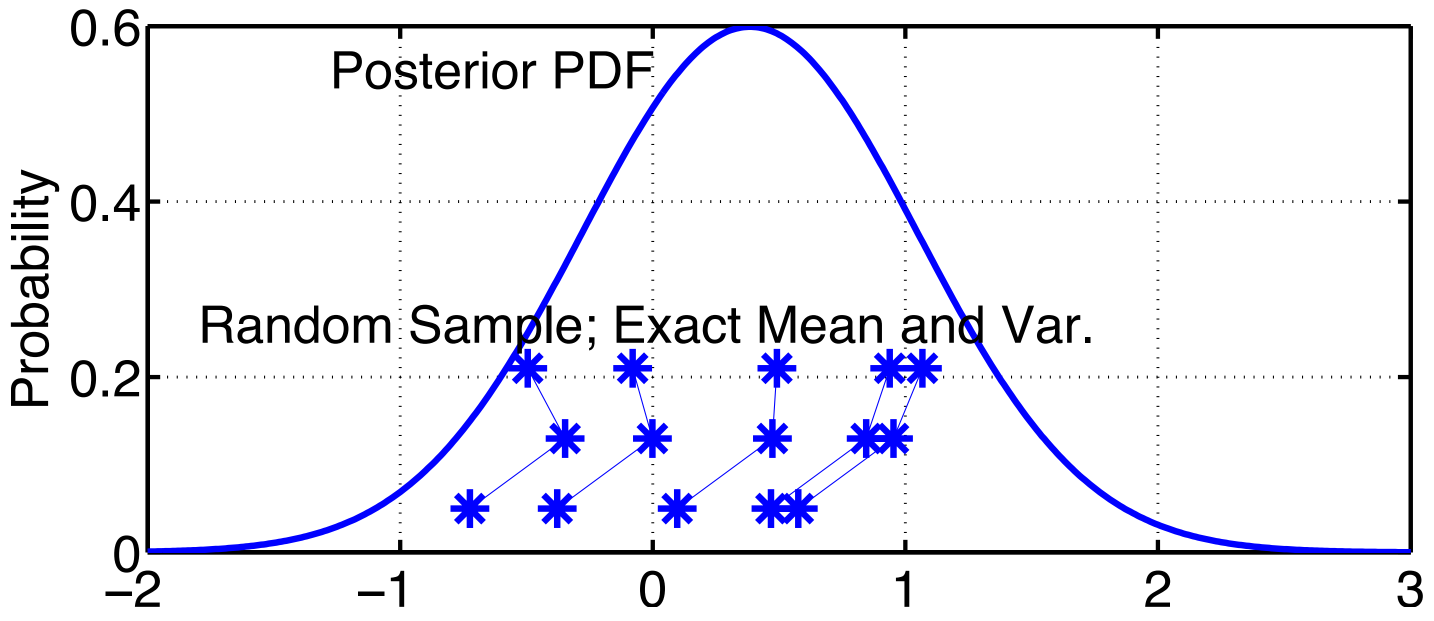


Can ‘play games’ with this sample to improve (modify) its properties.

Example: Adjust the mean of the sample to be exact.
Can also adjust the variance to be exact.

Sampling Posterior PDF:

Just draw a random sample

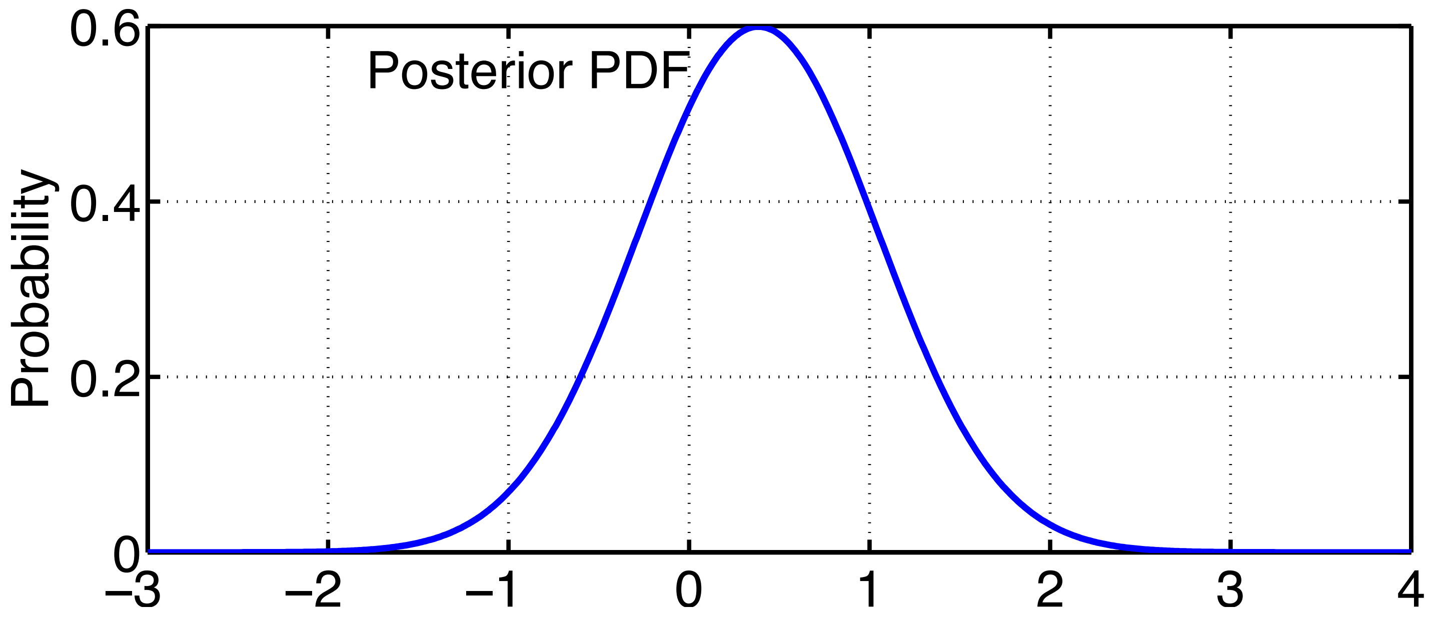


Might also want to eliminate rare extreme outliers.

NOTE: Properties of these adjusted samples can be quite different. How these properties interact with the rest of the assimilation is an open question.

Sampling Posterior PDF:

Construct a ‘deterministic’ sample with certain features.

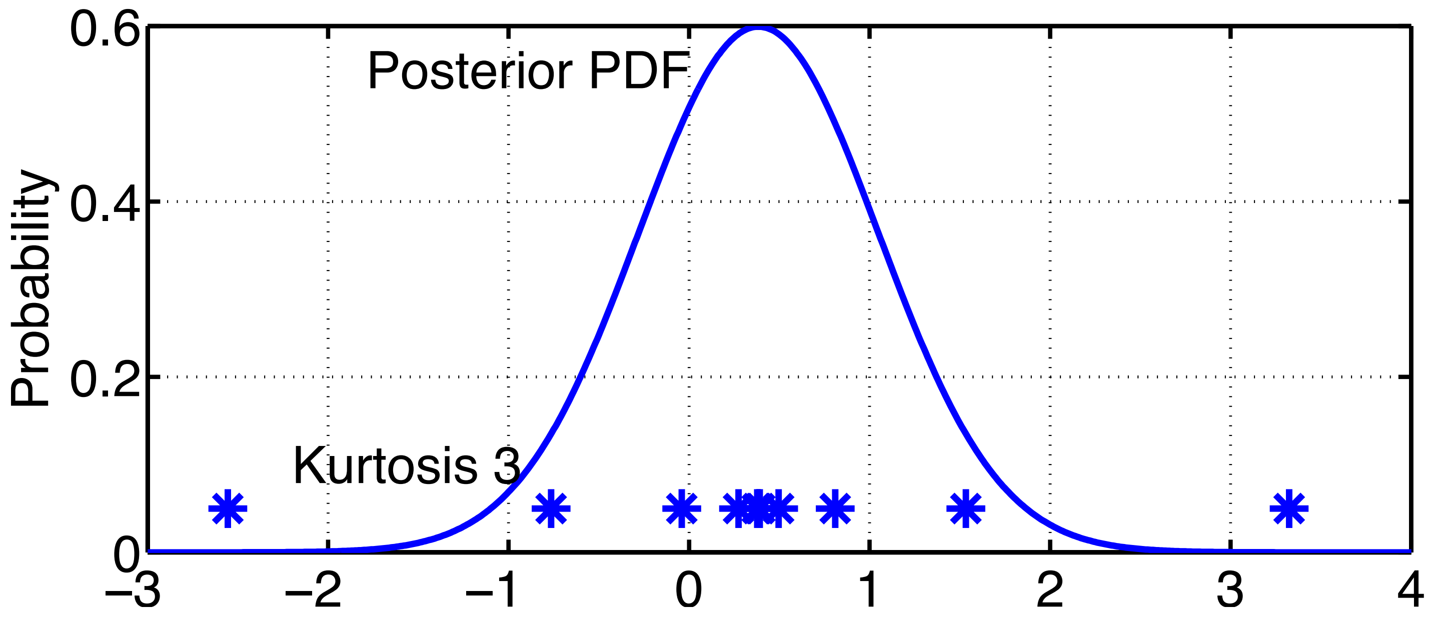


For instance: Sample could have exact mean and variance.

This is insufficient to constrain ensemble, need other constraints.

Sampling Posterior PDF:

Construct a 'deterministic' sample with certain features (*filter_kind*=6 in *assim_tools_nml*; manually adjust kurtosis).

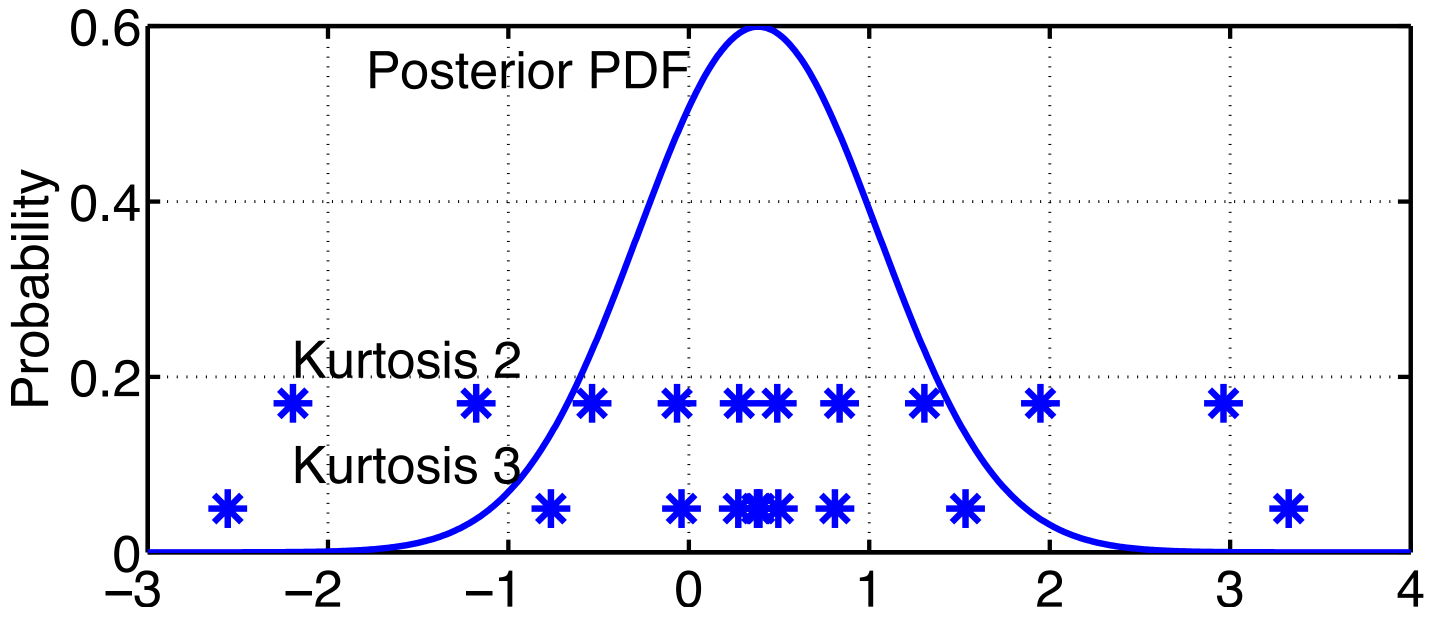


Example: Exact sample mean and variance.

Sample kurtosis is 3 (expected value for Gaussian in large sample limit) (Constructed by staring uniformly-spaced and adjusting quadratically).

Sampling Posterior PDF:

Construct a 'deterministic' sample with certain features (*filter_kind*=6 in *assim_tools_nml*; manually adjust kurtosis).

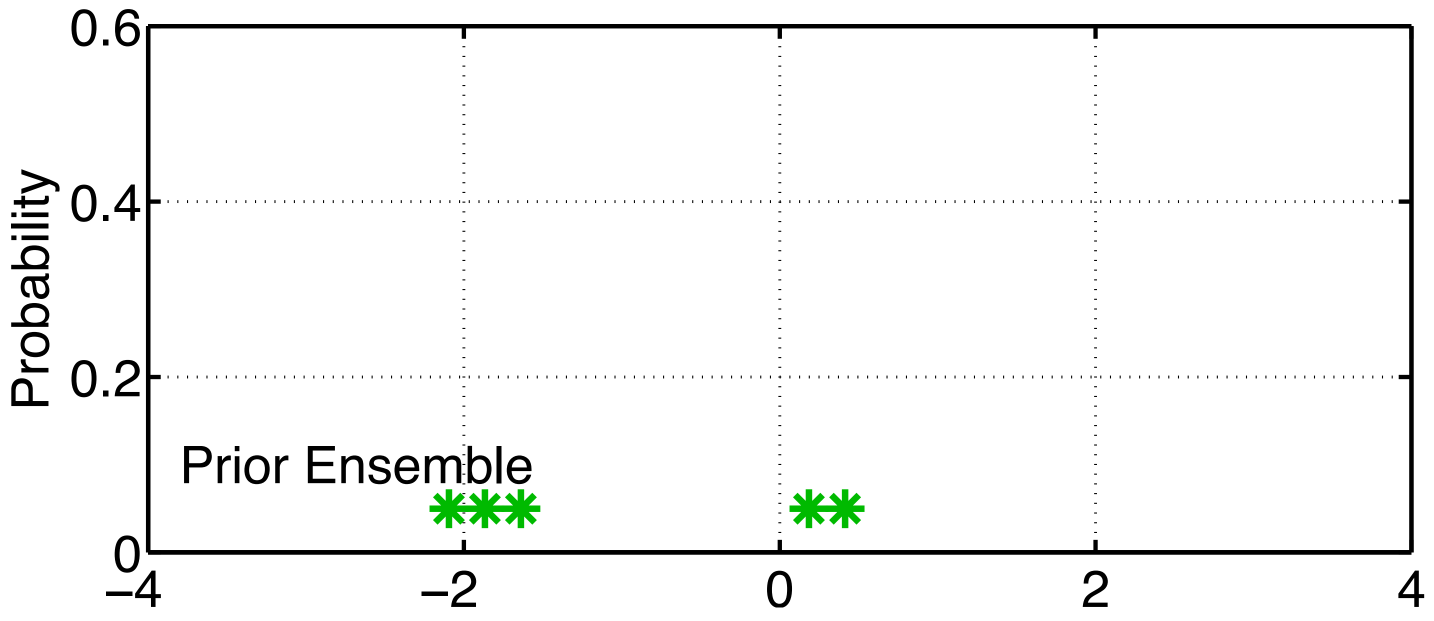


Example: Exact sample mean and variance.

Sample kurtosis 2: less extreme outliers, less dense near mean.
Avoiding outliers might be nice in certain applications.
Sampling heavily near mean might be nice.

Sampling Posterior PDF:

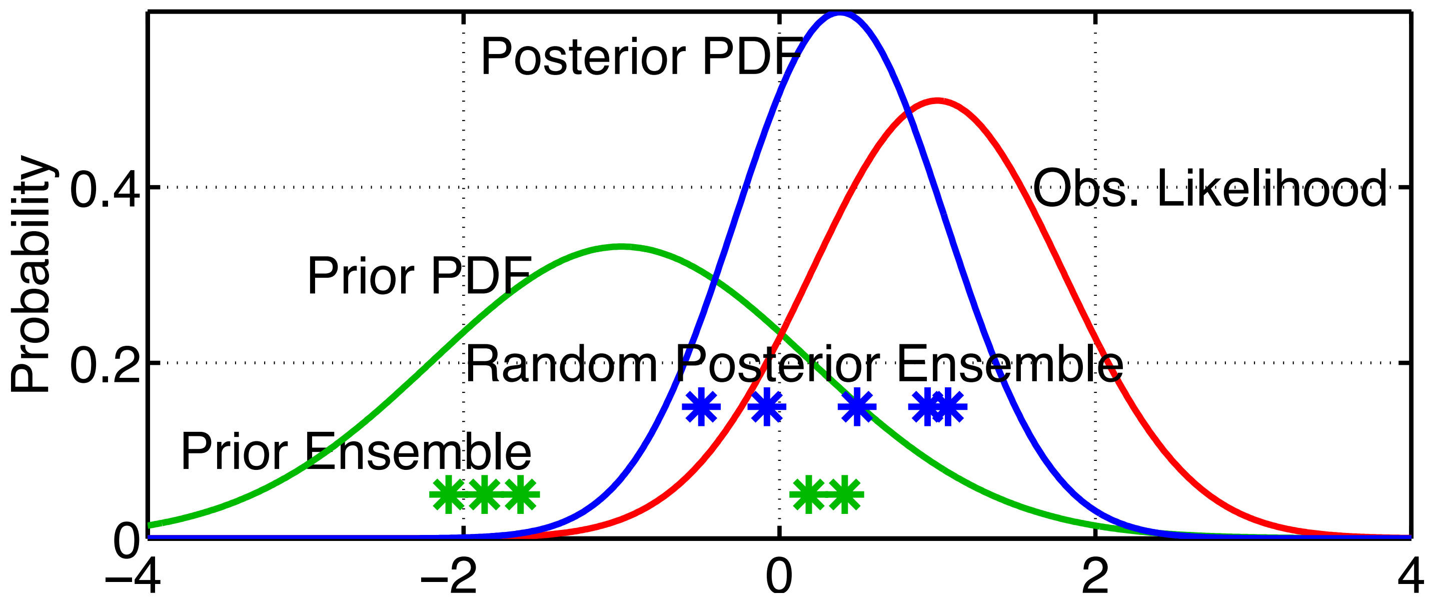
First two methods depend only on mean and variance of prior sample.



Example: Suppose prior sample is (significantly) bimodal?

Sampling Posterior PDF:

First two methods depend only on mean and variance of prior sample.

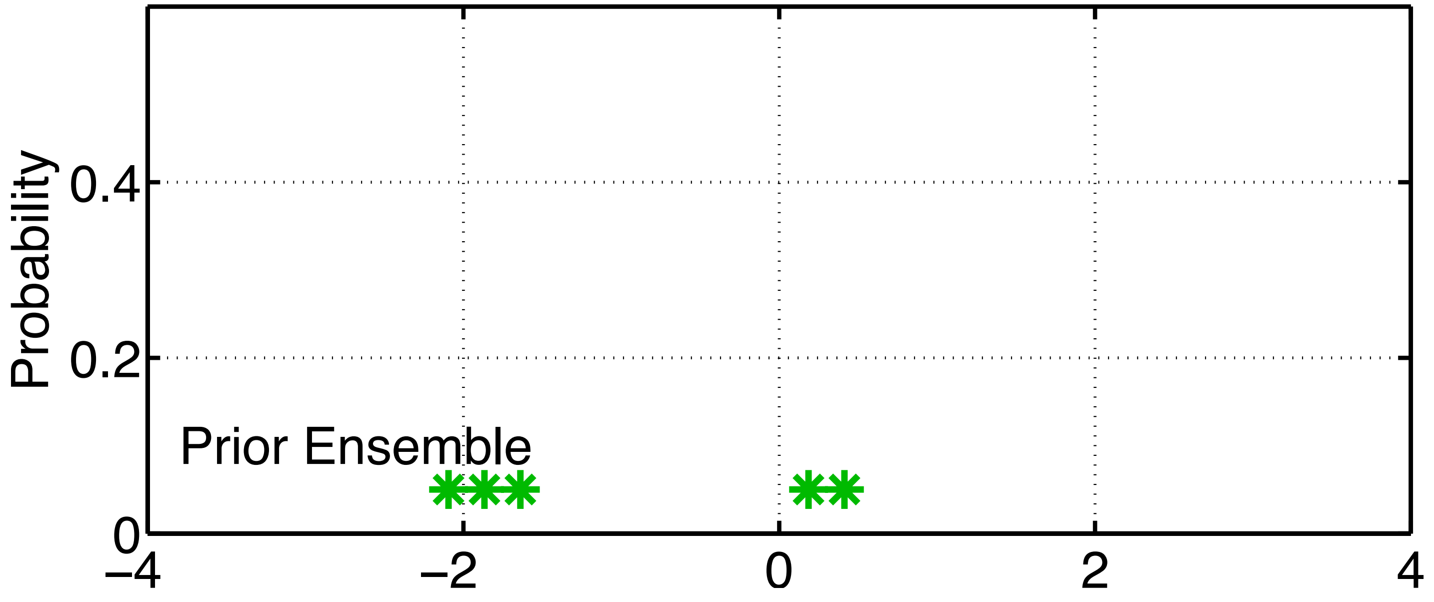


Example: Suppose prior sample is (significantly) bimodal?

Might want to retain additional information from prior.
Recall that Ensemble Adjustment Filter tried to do this (Section 1).

Ensemble Filter Algorithms:

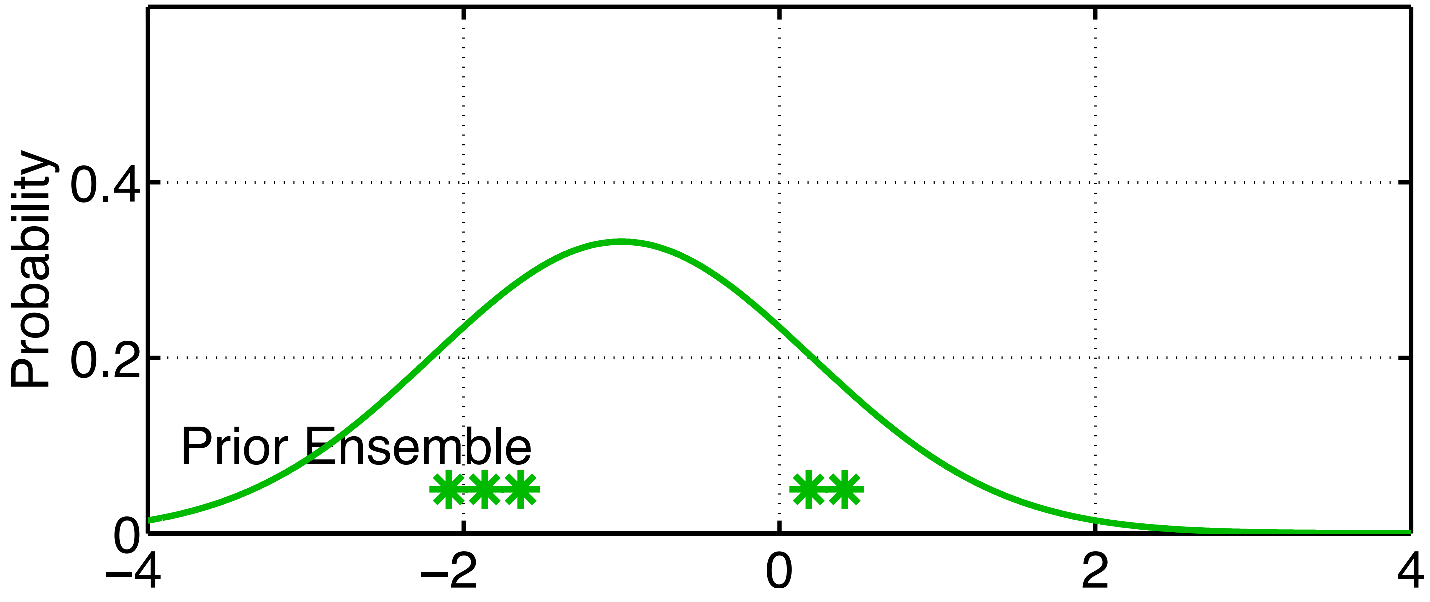
Ensemble Kalman Filter (EnKF) (*filter_kind*=2 in *assim_tools_nml*).



'Classical' Monte Carlo Algorithm for Data Assimilation.
Warning: earliest references have incorrect algorithm (more in a minute).

Ensemble Filter Algorithms:

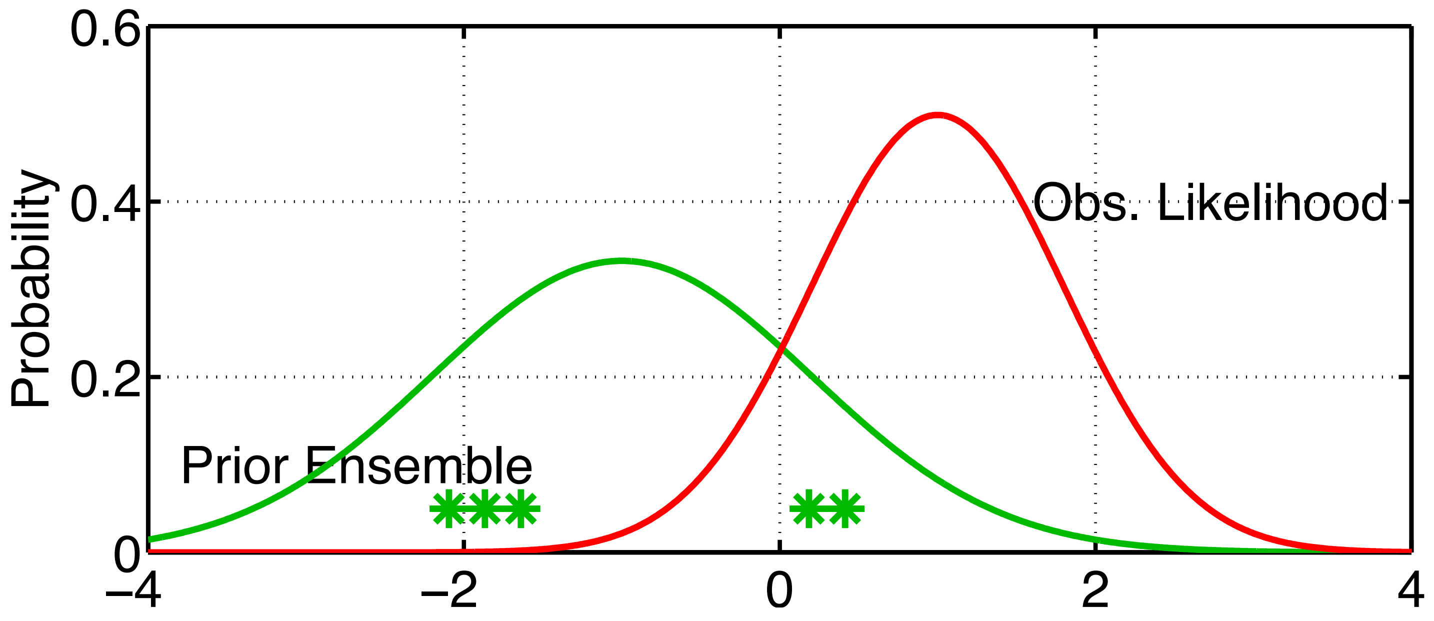
Ensemble Kalman Filter (EnKF) (*filter_kind*=2 in *assim_tools_nml*).



Again, fit a Gaussian to the sample.
Are there ways to do this without computing prior sample stats?

Ensemble Filter Algorithms:

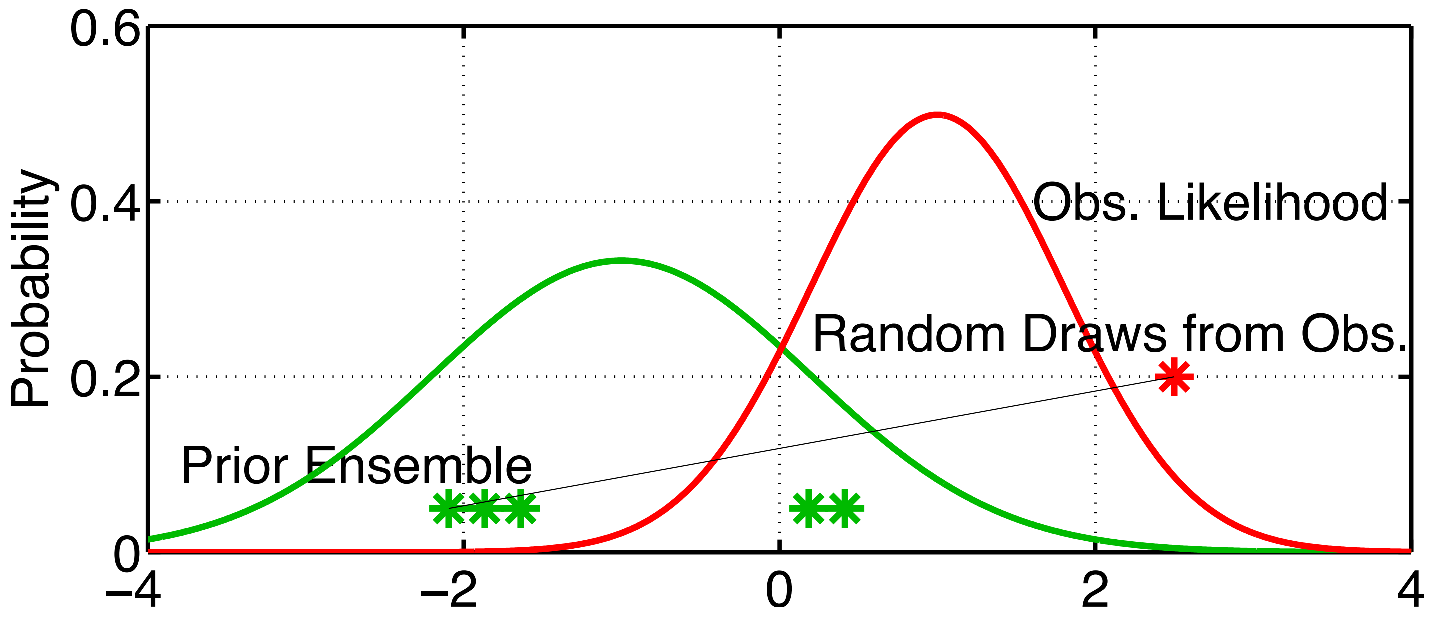
Ensemble Kalman Filter (EnKF) (*filter_kind*=2 in *assim_tools_nml*).



Again, fit a Gaussian to the sample.

Ensemble Filter Algorithms:

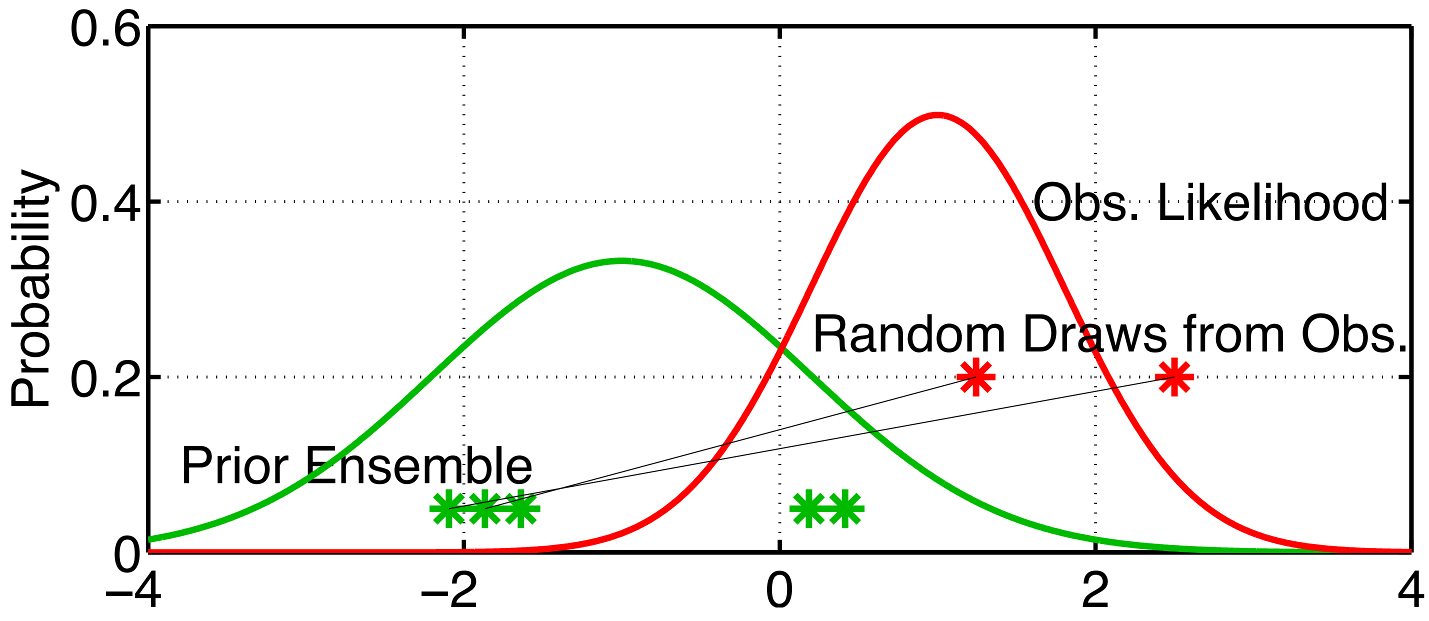
Ensemble Kalman Filter (EnKF) (*filter_kind*=2 in *assim_tools_nml*).



Generate a random draw from the observation likelihood.
Associate it with the first sample of the prior ensemble.

Ensemble Filter Algorithms:

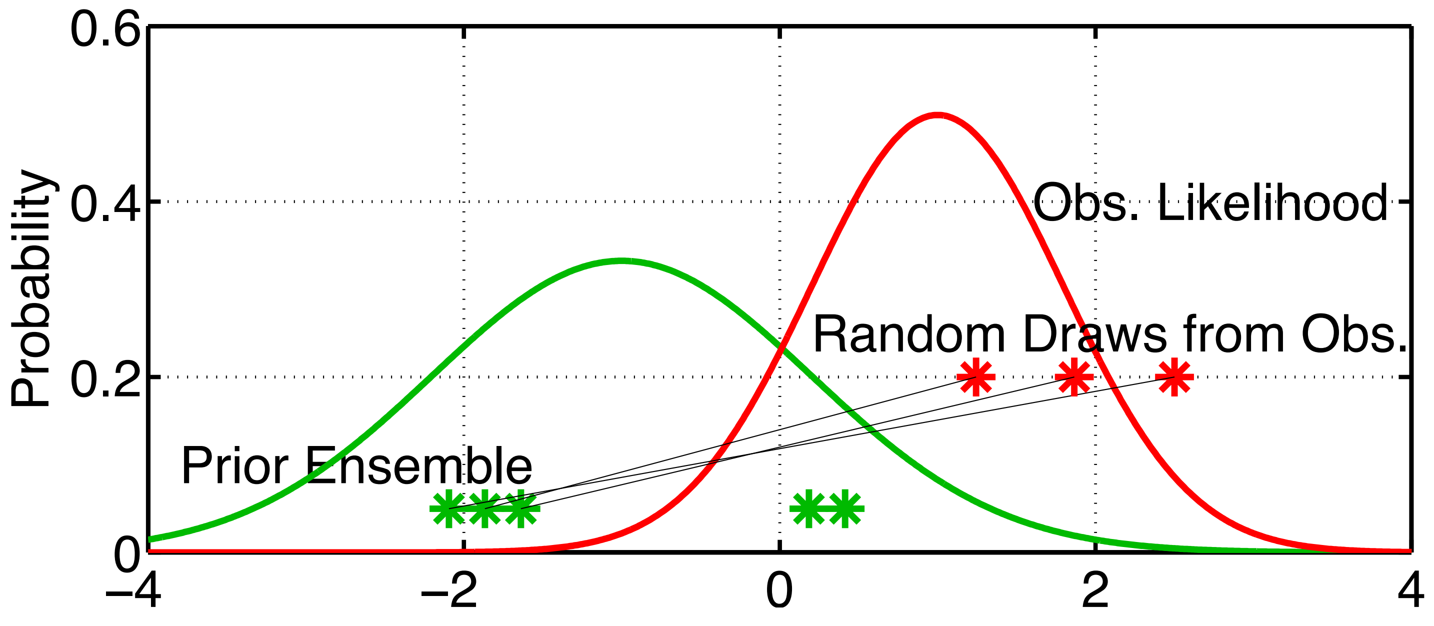
Ensemble Kalman Filter (EnKF) (*filter_kind*=2 in *assim_tools_nml*).



Proceed to associate a random draw from obs. with each prior sample.
This has been called 'perturbed' observations.
Algorithm sometimes called 'perturbed obs.' ensemble Kalman filter.

Ensemble Filter Algorithms:

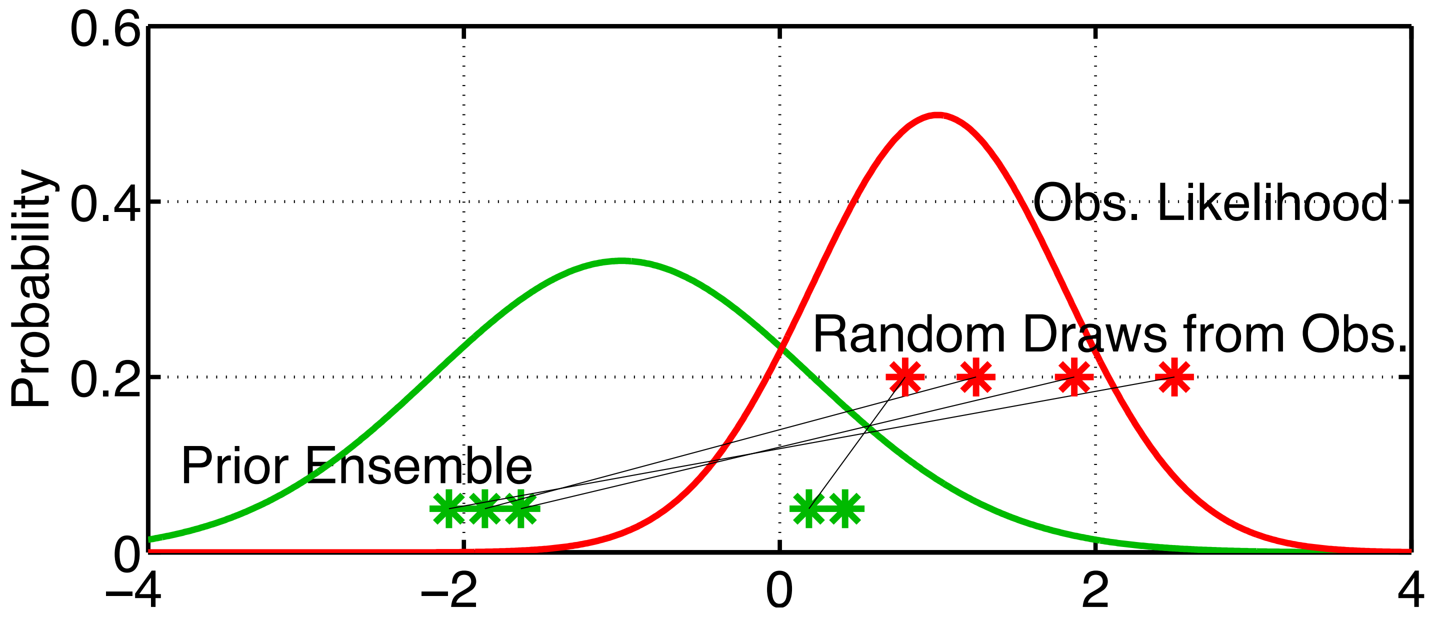
Ensemble Kalman Filter (EnKF) (*filter_kind*=2 in *assim_tools_nml*).



Proceed to associate a random draw from obs. with each prior sample.
This has been called ‘perturbed’ observations.
Algorithm sometimes called ‘perturbed obs.’ ensemble Kalman filter.

Ensemble Filter Algorithms:

Ensemble Kalman Filter (EnKF) (*filter_kind*=2 in *assim_tools_nml*).



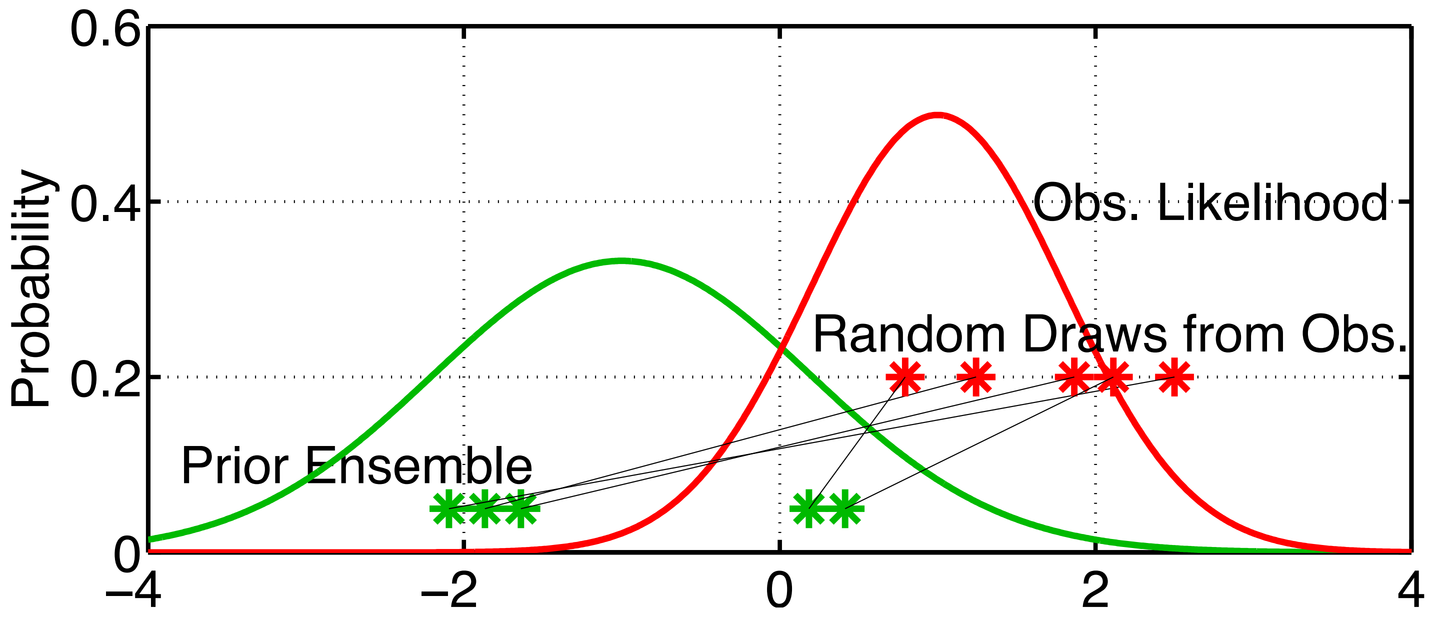
Proceed to associate a random draw from obs. with each prior sample.

Earliest publications associated mean of obs. likelihood with each prior.

This resulted in insufficient variance in posterior.

Ensemble Filter Algorithms:

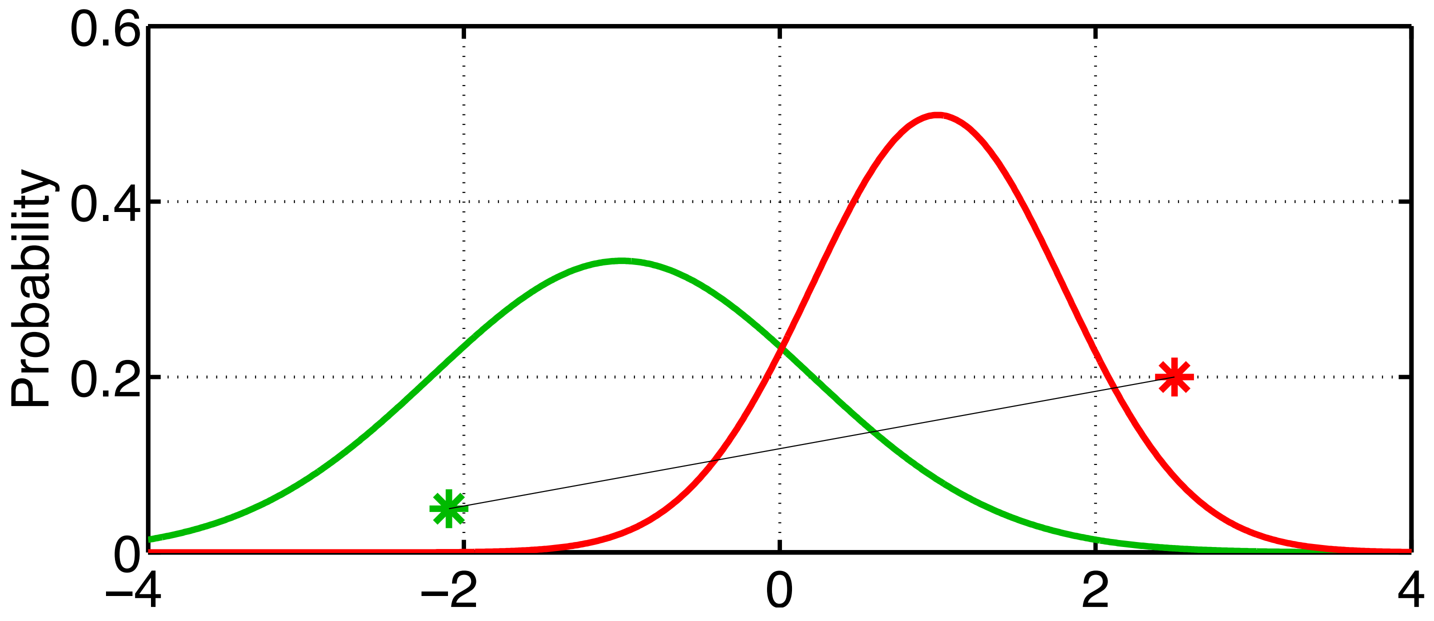
Ensemble Kalman Filter (EnKF) (*filter_kind*=2 in *assim_tools_nml*).



Have sample of joint prior distribution for observation and prior MEAN.
Adjusting the mean of obs. sample to be exact improves performance.
Adjusting the variance may further improve performance.
Outliers are potential problem, but can be removed.

Ensemble Filter Algorithms:

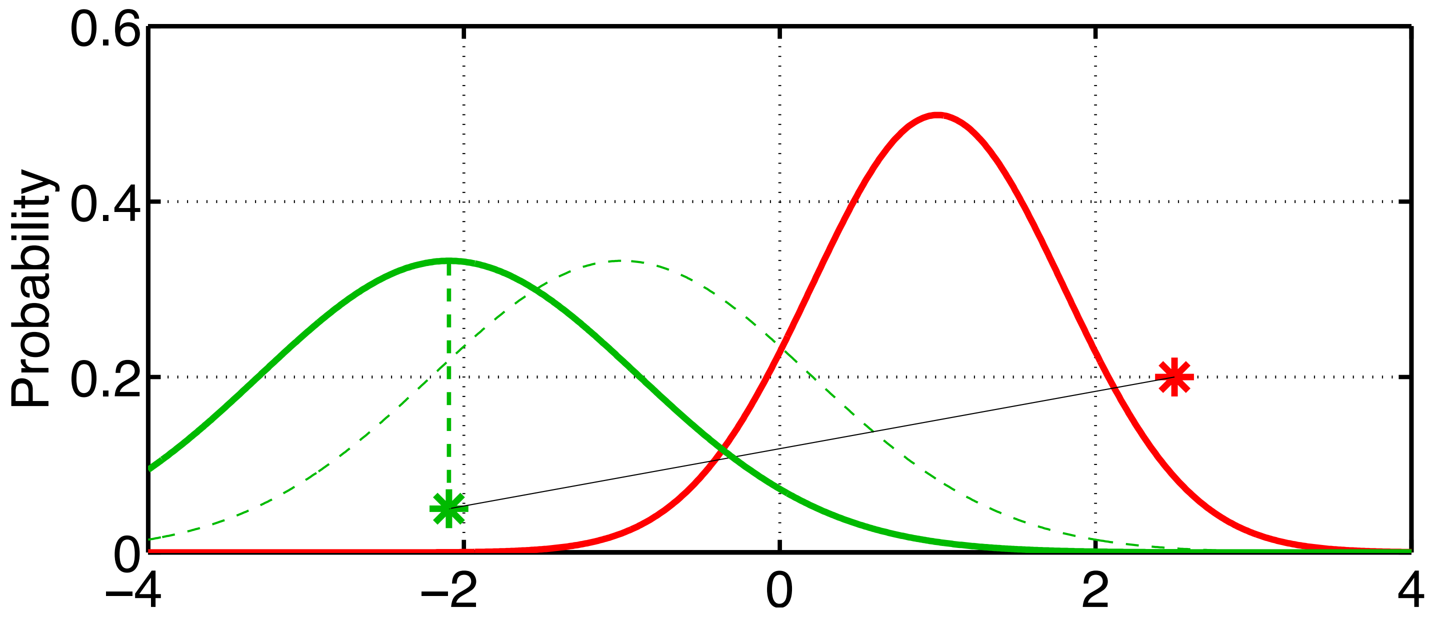
Ensemble Kalman Filter (EnKF) (*filter_kind*=2 in *assim_tools_nml*).



For each prior mean/obs. pair, find mean of posterior PDF.

Ensemble Filter Algorithms:

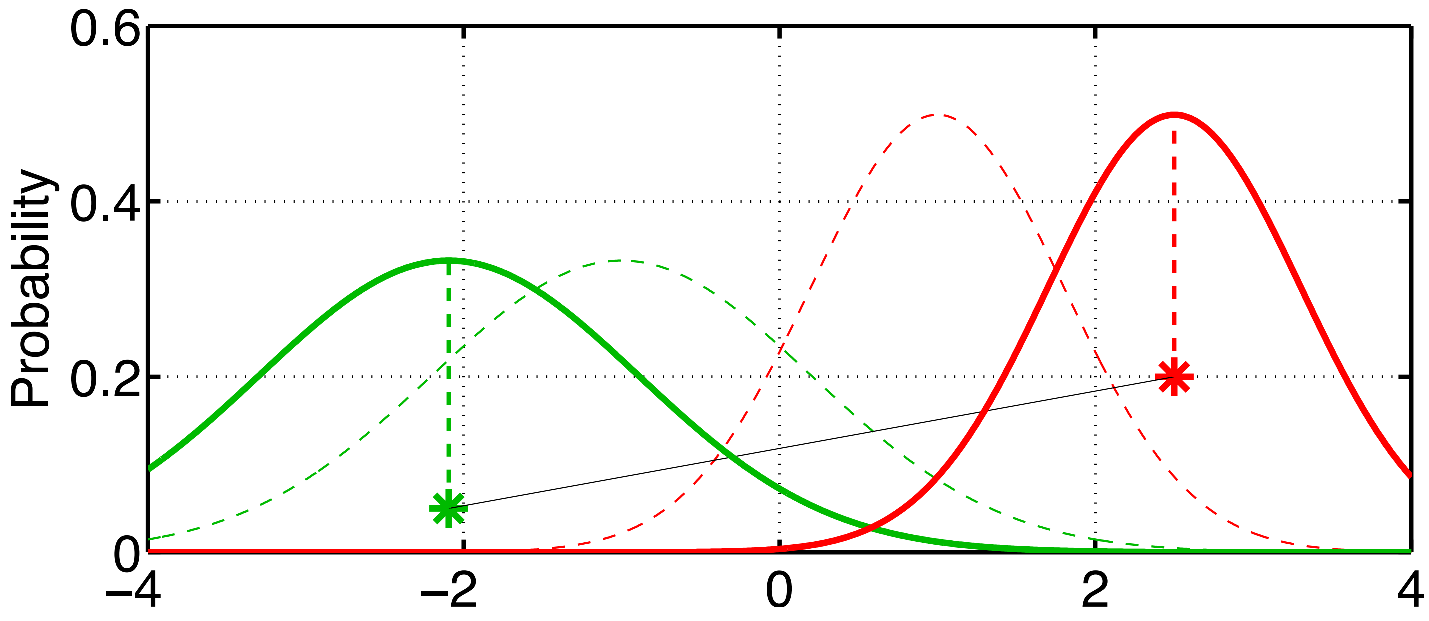
Ensemble Kalman Filter (EnKF) (*filter_kind*=2 in *assim_tools_nml*).



Prior sample standard deviation still measures uncertainty of prior mean estimate.

Ensemble Filter Algorithms:

Ensemble Kalman Filter (EnKF) (*filter_kind*=2 in *assim_tools_nml*).

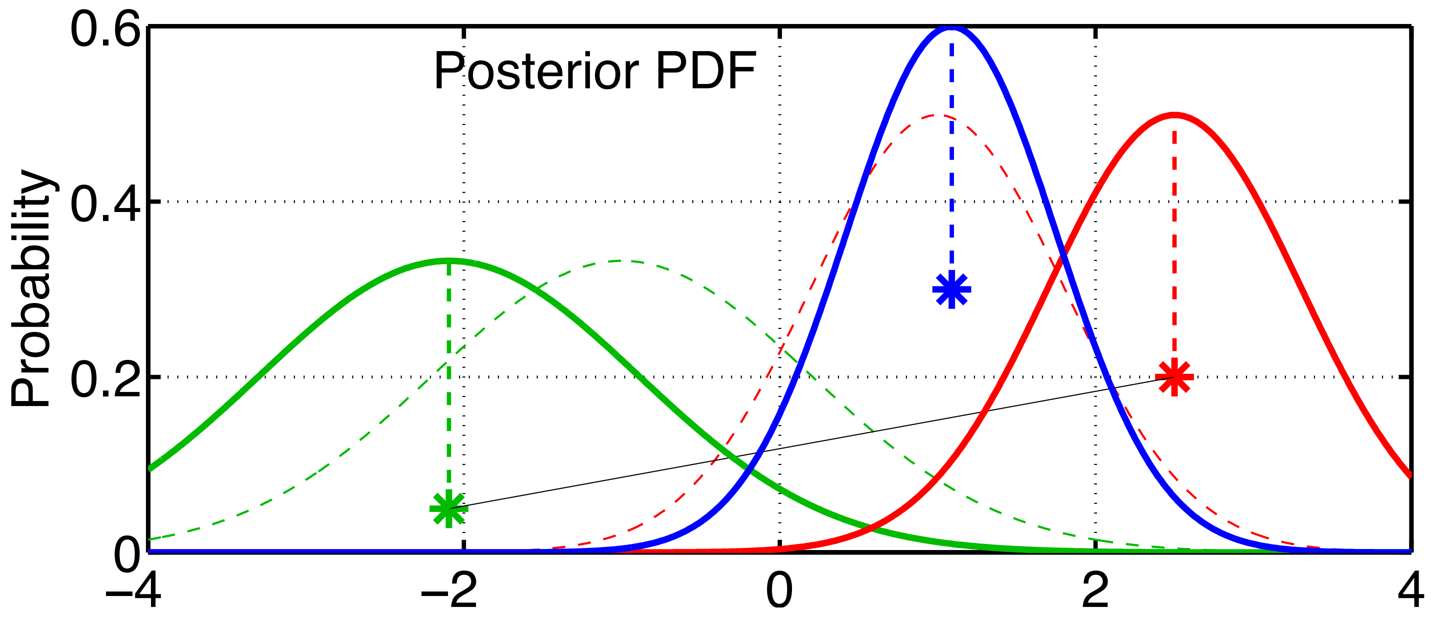


Prior sample standard deviation still measures uncertainty of prior mean estimate.

Obs. likelihood standard deviation measures uncertainty of obs. estimate.

Ensemble Filter Algorithms:

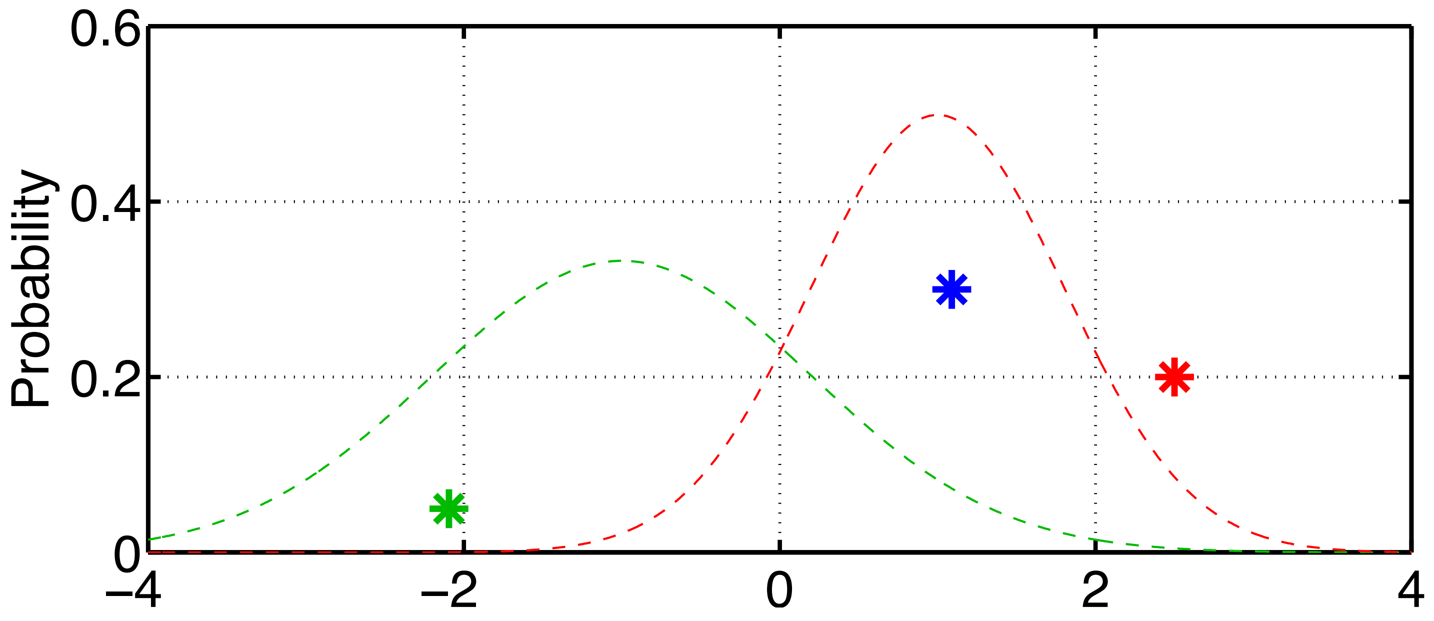
Ensemble Kalman Filter (EnKF) (*filter_kind*=2 in *assim_tools_nml*).



Take product of the prior/obs distributions for first sample.
This is the standard Gaussian product.

Ensemble Filter Algorithms:

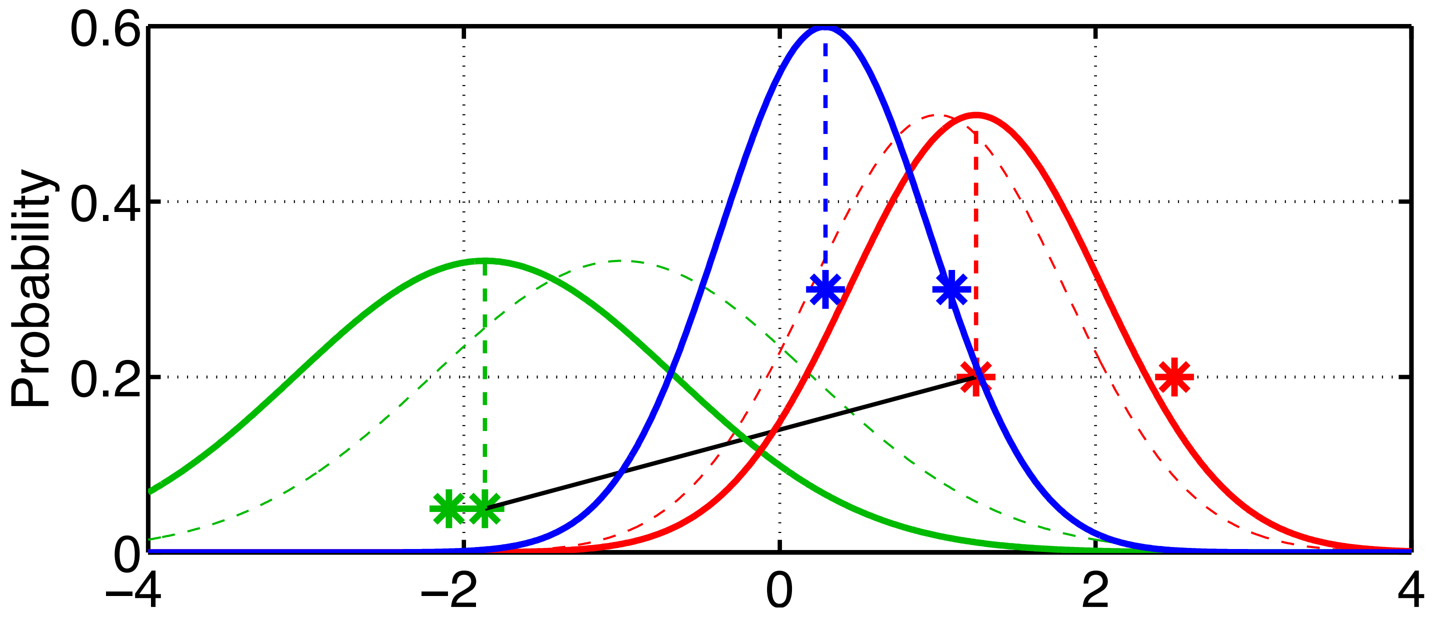
Ensemble Kalman Filter (EnKF) (*filter_kind*=2 in *assim_tools_nml*).



Mean of product is random sample of posterior.
Product of random samples is random sample of product.

Ensemble Filter Algorithms:

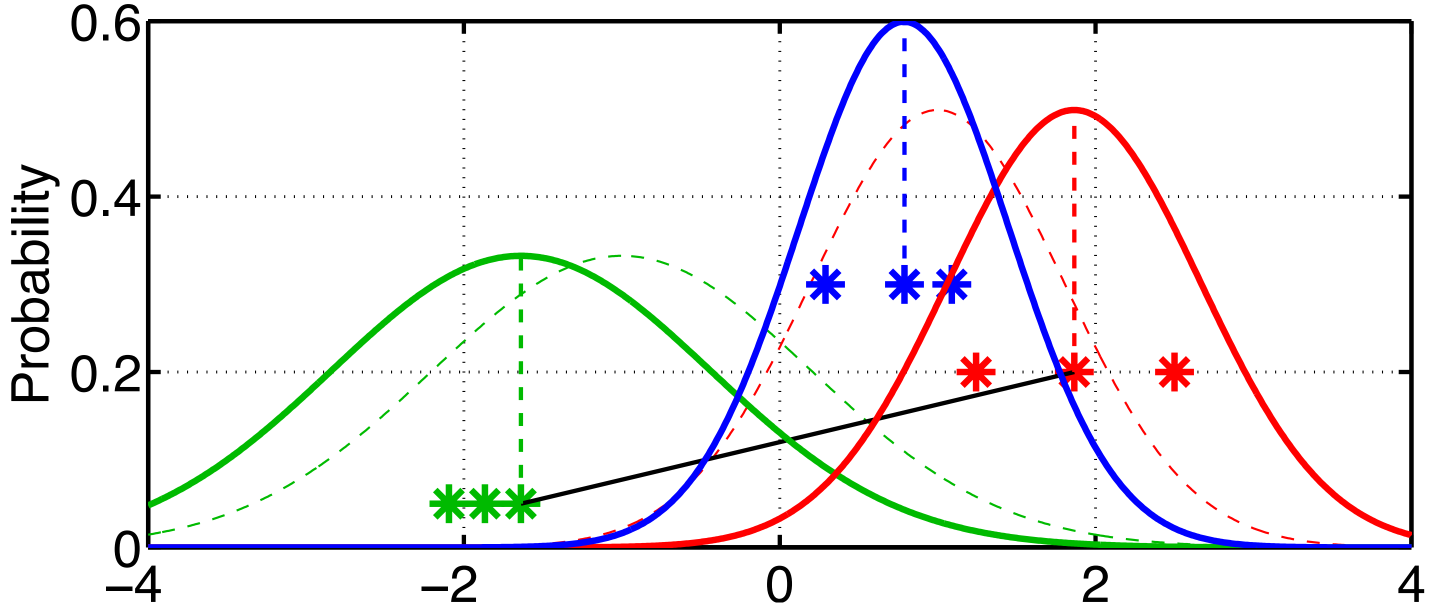
Ensemble Kalman Filter (EnKF) (*filter_kind*=2 in *assim_tools_nml*).



Repeat this operation for each joint prior pair.

Ensemble Filter Algorithms:

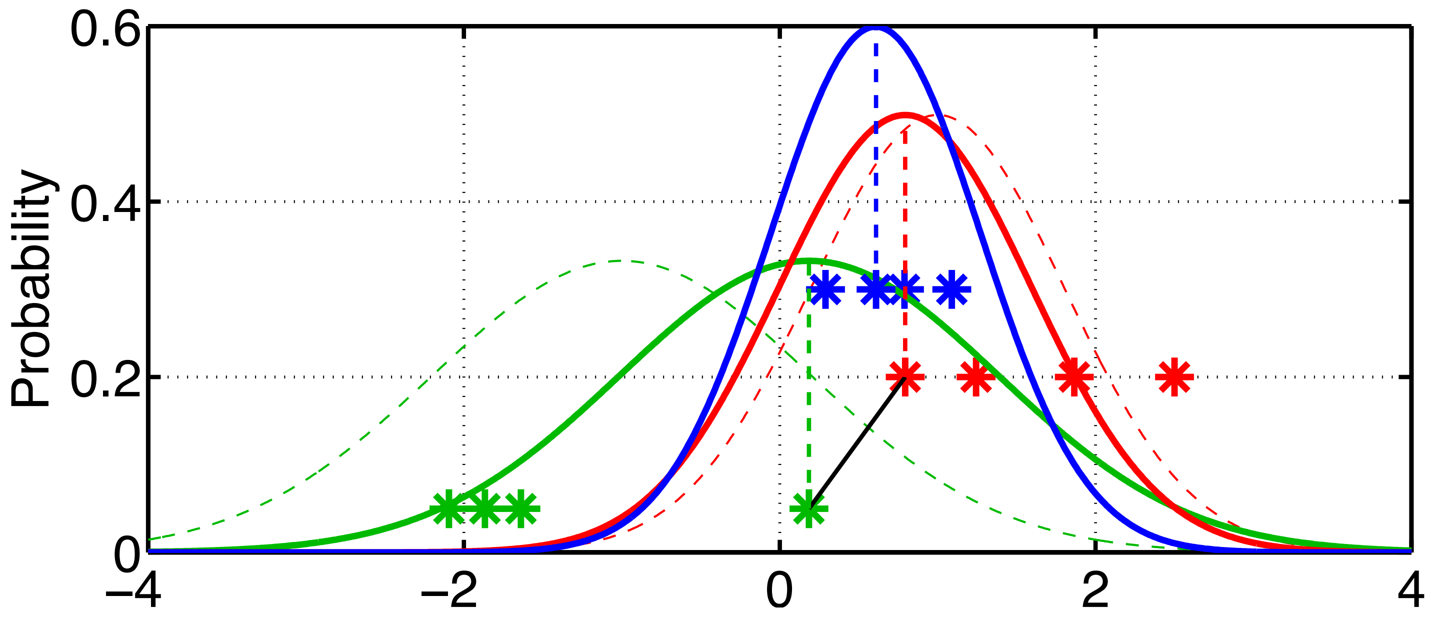
Ensemble Kalman Filter (EnKF) (*filter_kind*=2 in *assim_tools_nml*).



Repeat this operation for each joint prior pair.

Ensemble Filter Algorithms:

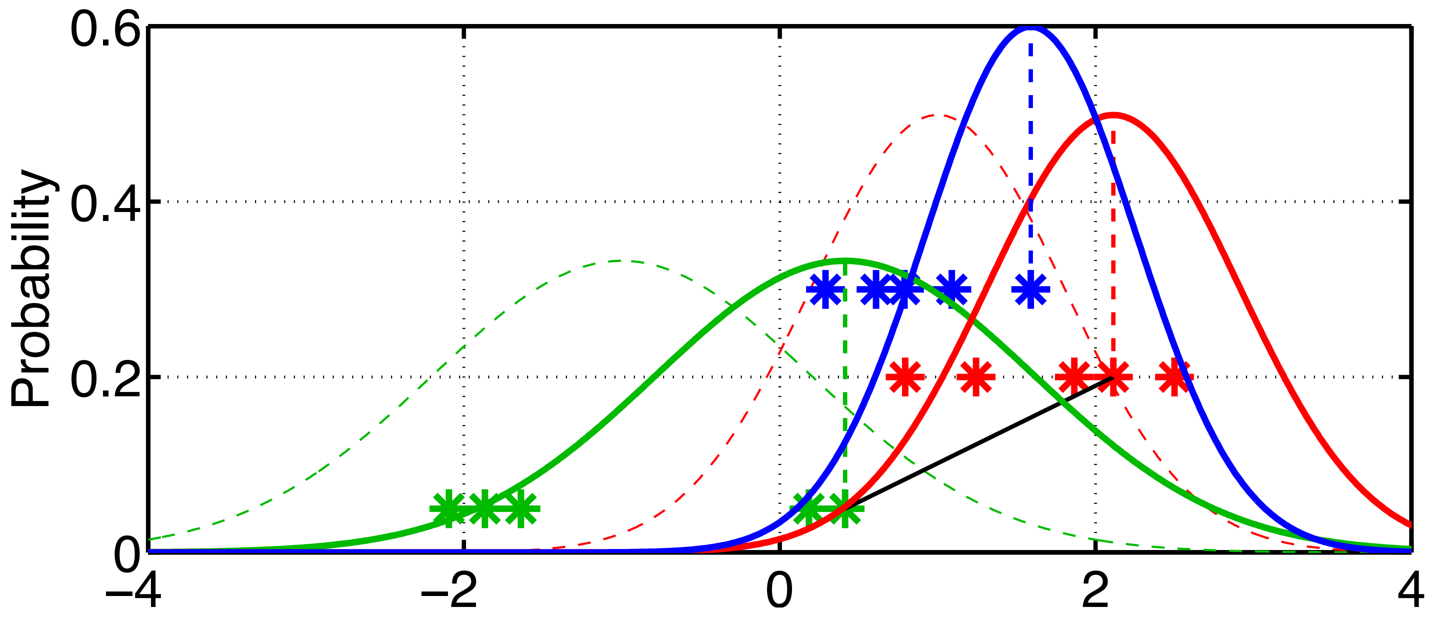
Ensemble Kalman Filter (EnKF) (*filter_kind*=2 in *assim_tools_nml*).



Repeat this operation for each joint prior pair.

Ensemble Filter Algorithms:

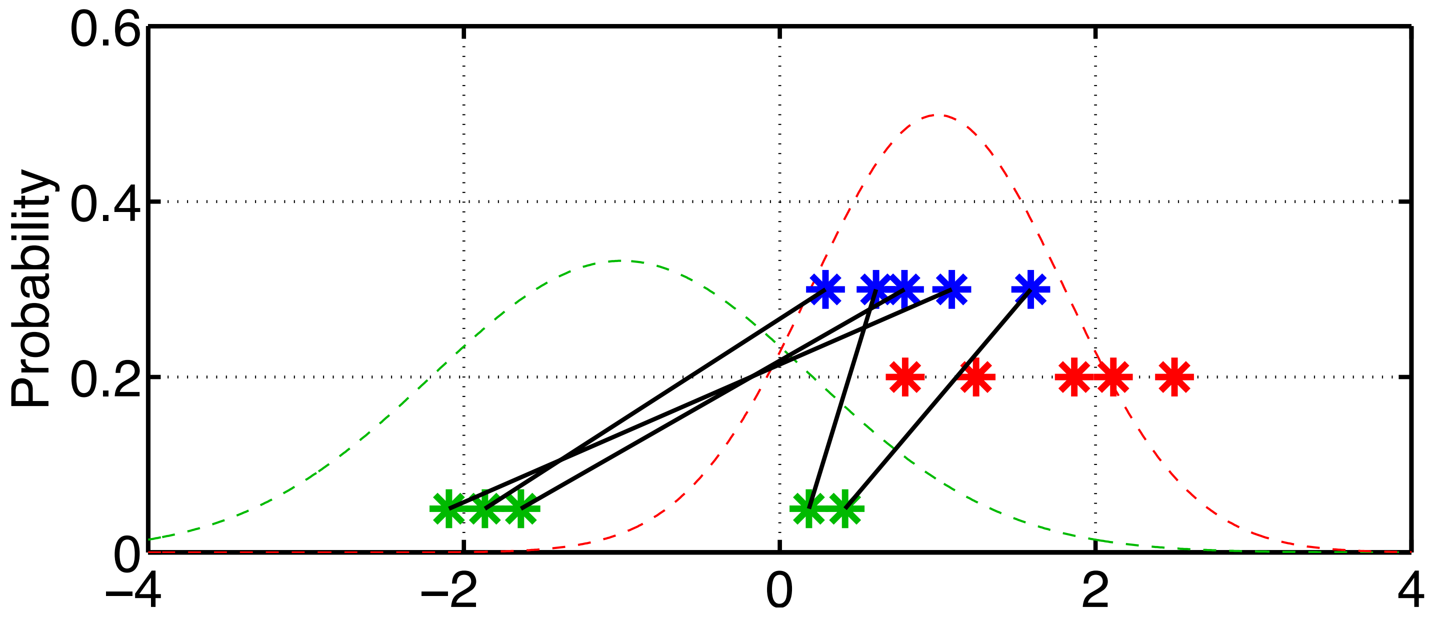
Ensemble Kalman Filter (EnKF) (*filter_kind*=2 in *assim_tools_nml*).



Repeat this operation for each joint prior pair.

Ensemble Filter Algorithms:

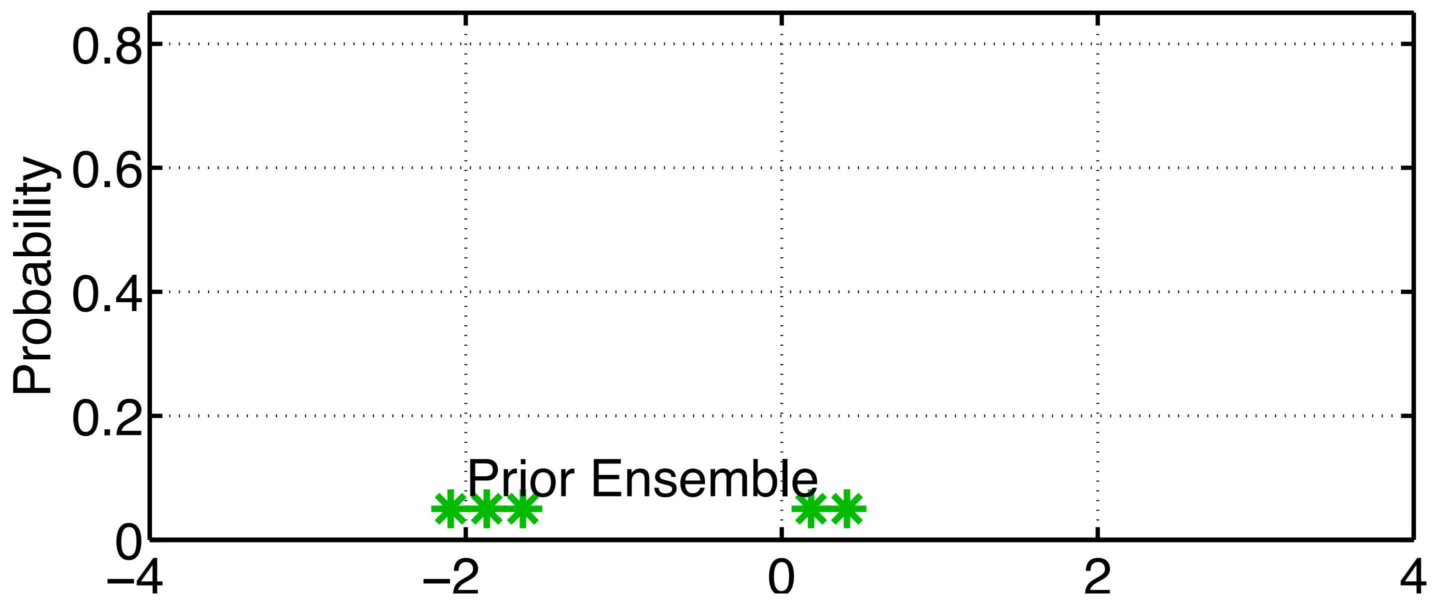
Ensemble Kalman Filter (EnKF) (*filter_kind*=2 in *assim_tools_nml*).



Posterior sample maintains much of prior sample structure.
(This is more apparent for larger ensemble sizes.)
Posterior sample mean and variance converge to ‘exact’ for large samples.

Ensemble Filter Algorithms:

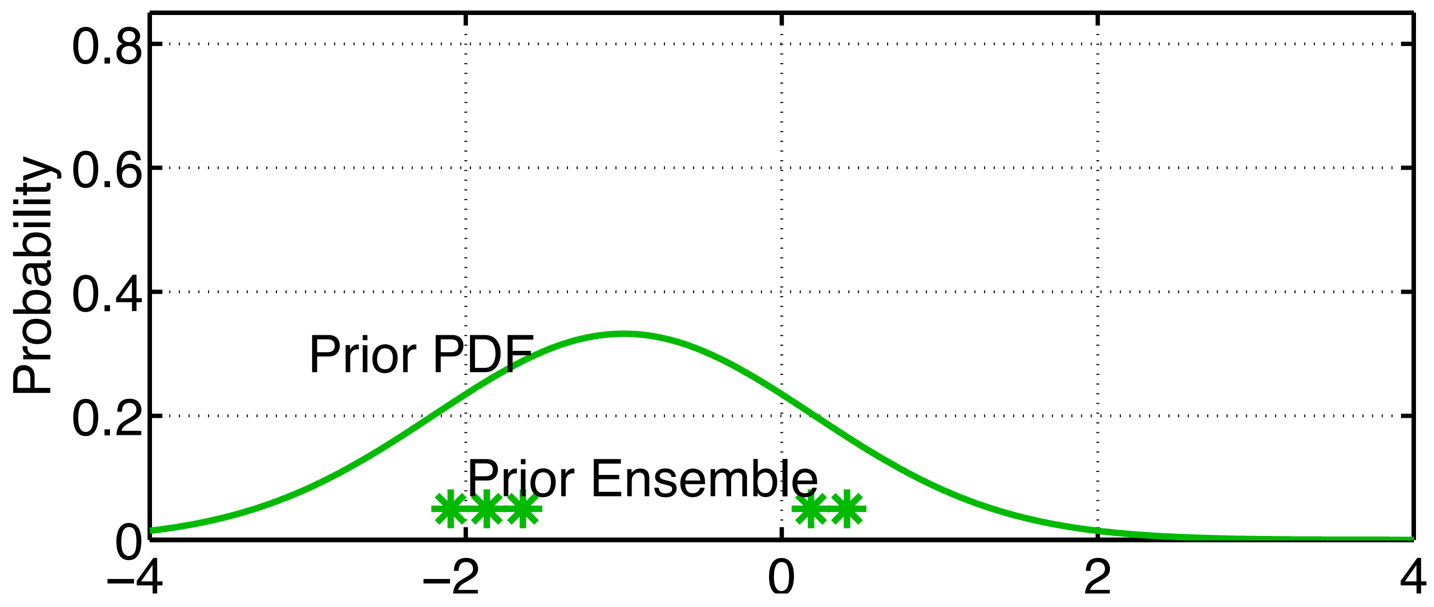
Ensemble Kernel Filter (EKF) (*filter_kind*=3 in *assim_tools_nml*).



Can retain more correct information about non-Gaussian priors.
Can also be used for obs. likelihood term in product (not shown here).

Ensemble Filter Algorithms:

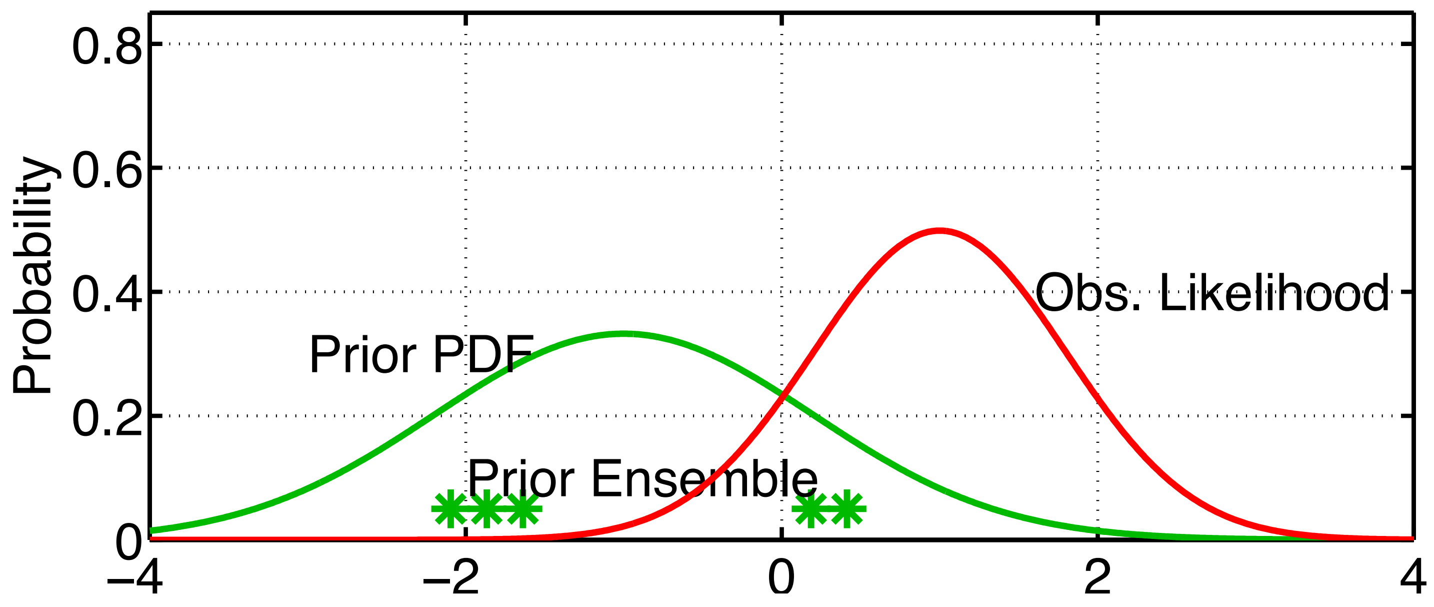
Ensemble Kernel Filter (EKF) (*filter_kind*=3 in *assim_tools_nml*).



Usually, kernel widths are a function of the sample variance.
Almost avoids using prior sample variance.

Ensemble Filter Algorithms:

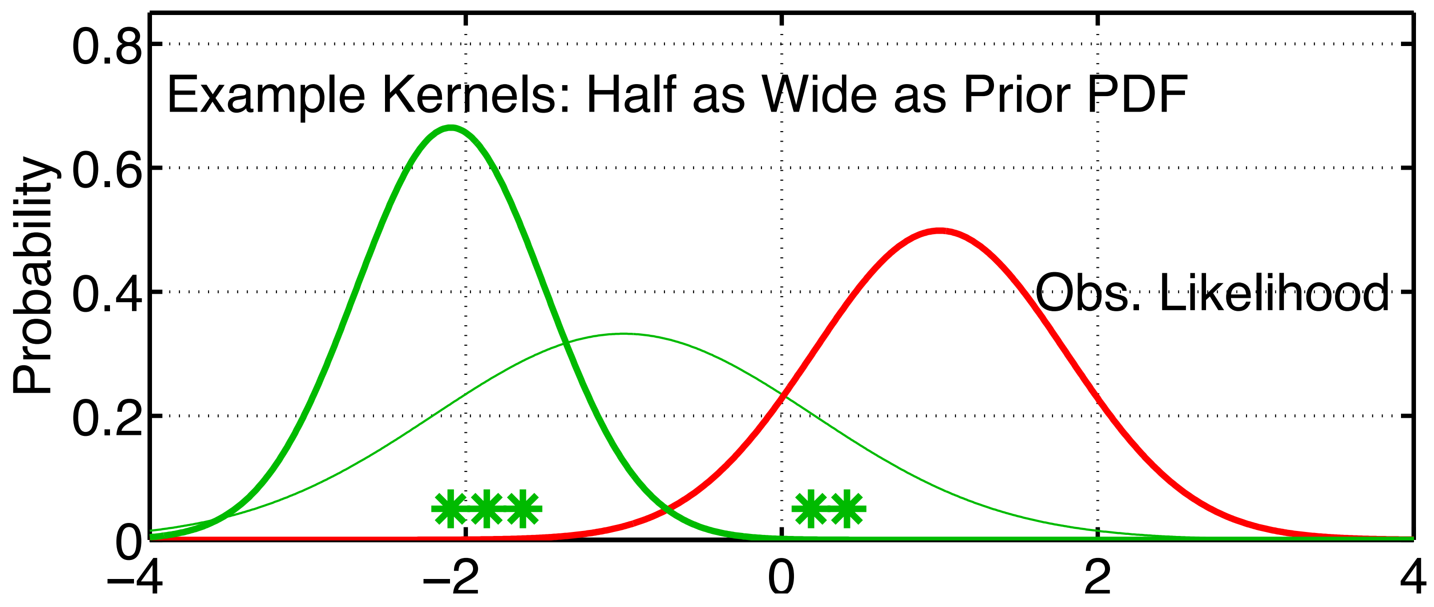
Ensemble Kernel Filter (EKF) (*filter_kind*=3 in *assim_tools_nml*).



Usually, kernel widths are a function of the sample variance.
Almost avoids using prior sample variance.

Ensemble Filter Algorithms:

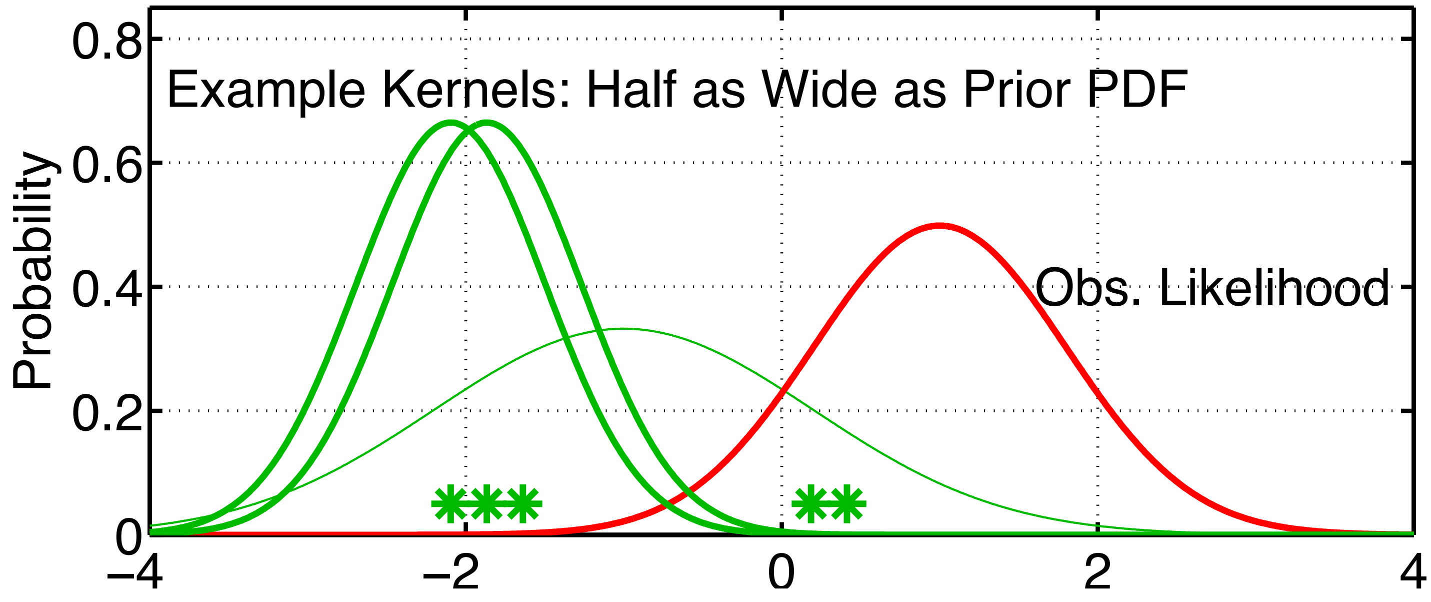
Ensemble Kernel Filter (EKF) (*filter_kind*=3 in *assim_tools_nml*).



Approximate prior as a sum of Gaussians centered on each sample.

Ensemble Filter Algorithms:

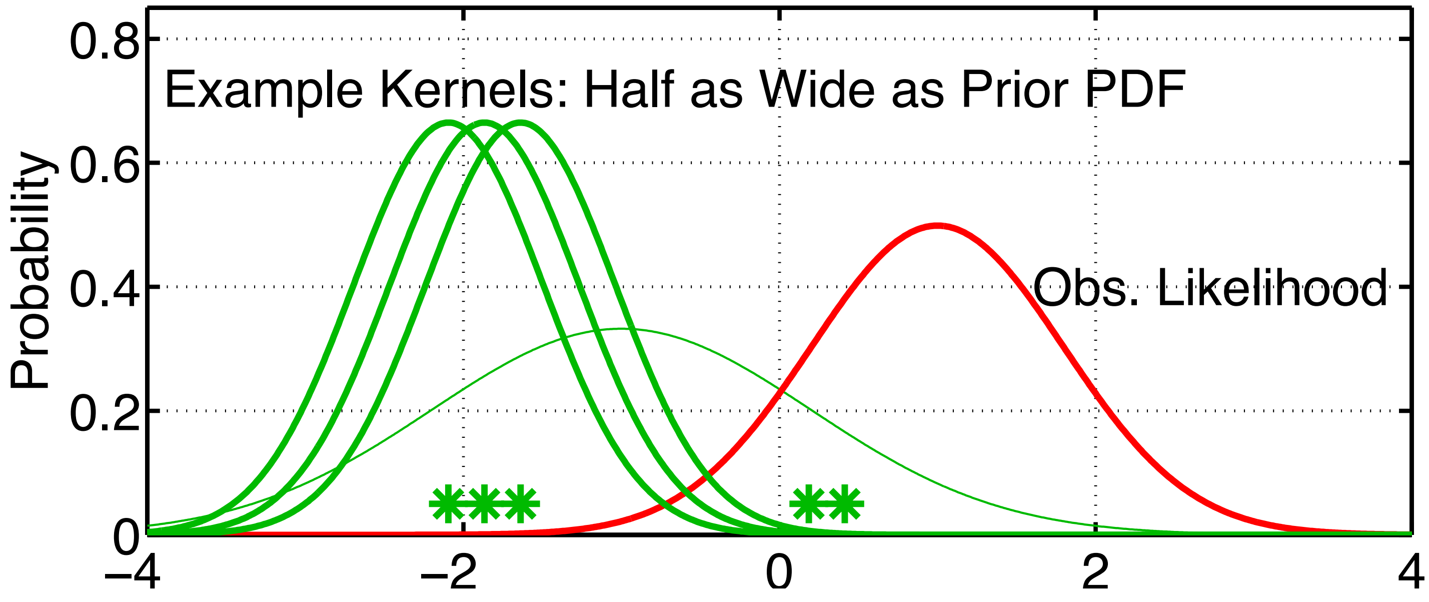
Ensemble Kernel Filter (EKF) (*filter_kind*=3 in *assim_tools_nml*).



Approximate prior as a sum of Gaussians centered on each sample.

Ensemble Filter Algorithms:

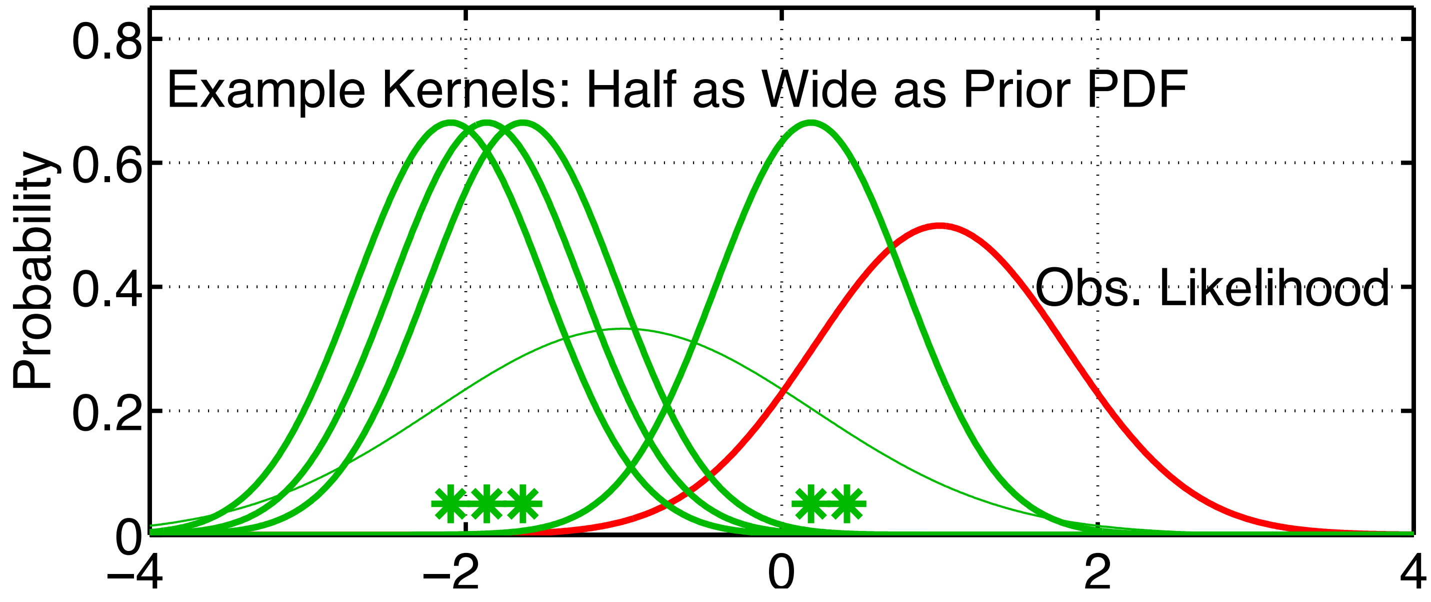
Ensemble Kernel Filter (EKF) (*filter_kind*=3 in *assim_tools_nml*).



Approximate prior as a sum of Gaussians centered on each sample.

Ensemble Filter Algorithms:

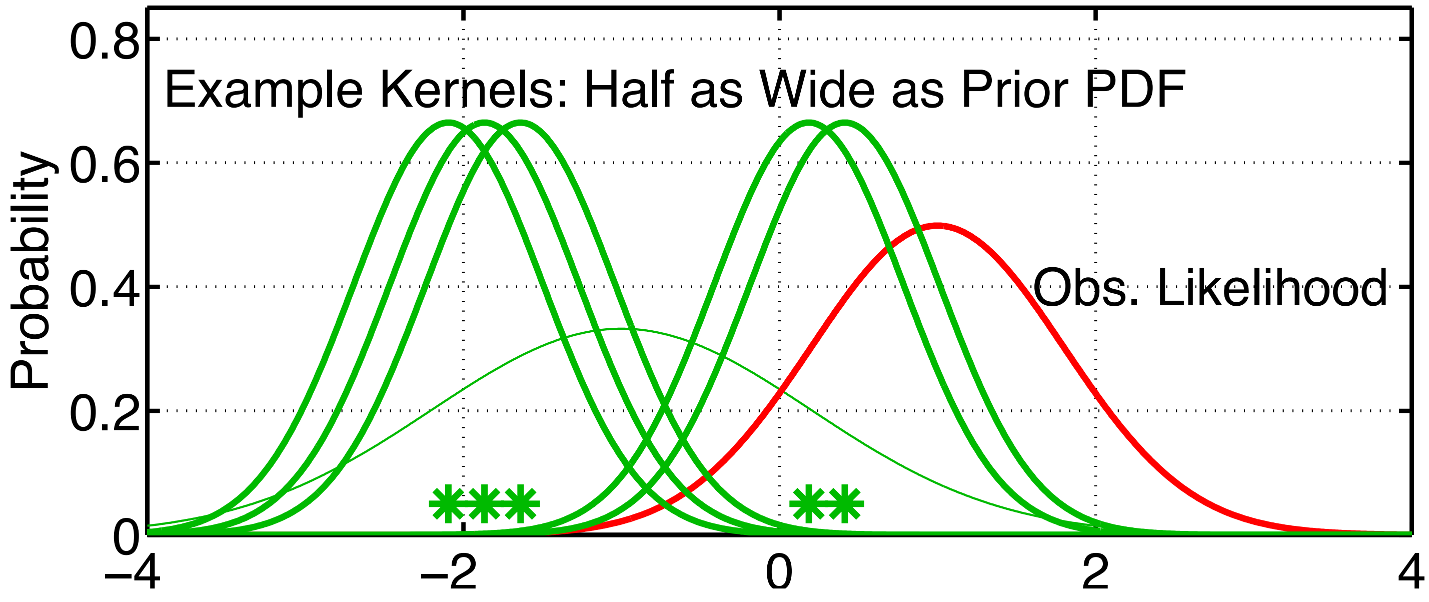
Ensemble Kernel Filter (EKF) (*filter_kind*=3 in *assim_tools_nml*).



Approximate prior as a sum of Gaussians centered on each sample.

Ensemble Filter Algorithms:

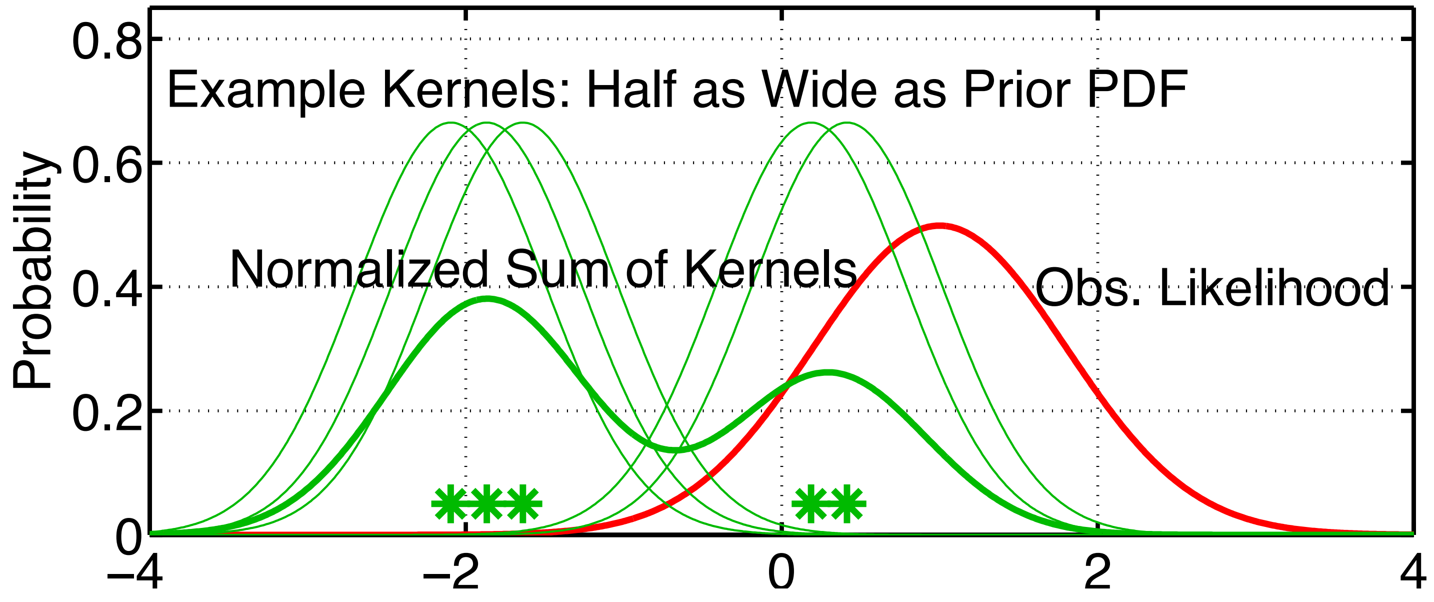
Ensemble Kernel Filter (EKF) (*filter_kind*=3 in *assim_tools_nml*).



Approximate prior as a sum of Gaussians centered on each sample.

Ensemble Filter Algorithms:

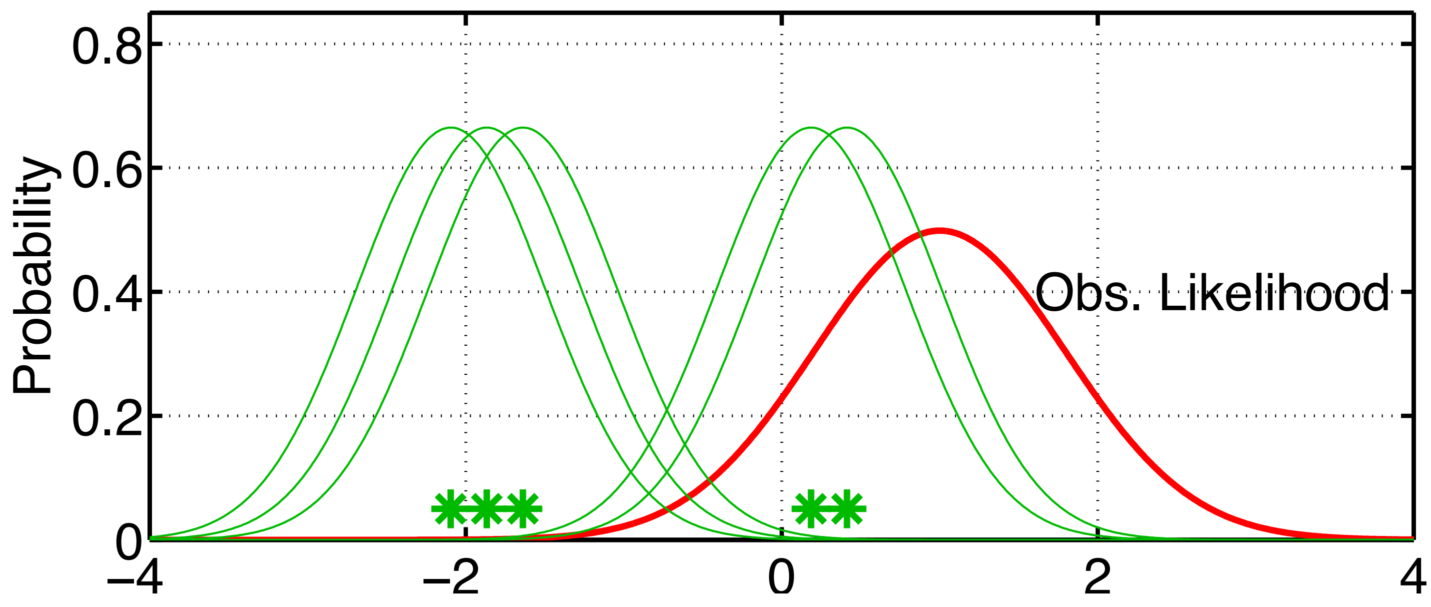
Ensemble Kernel Filter (EKF) (*filter_kind*=3 in *assim_tools_nml*).



The estimate of the prior is the normalized sum of all kernels.

Ensemble Filter Algorithms:

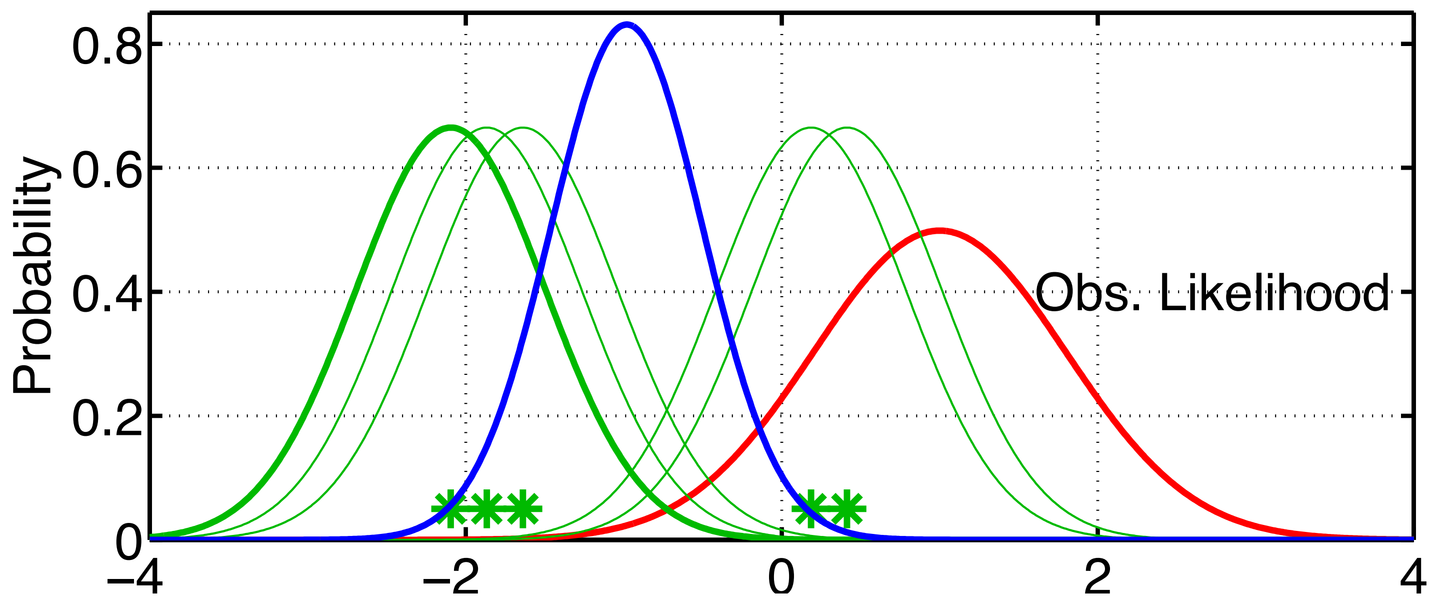
Ensemble Kernel Filter (EKF) (*filter_kind*=3 in *assim_tools_nml*).



Apply distributive law to take product.
Product of the sum is the sum of the products.

Ensemble Filter Algorithms:

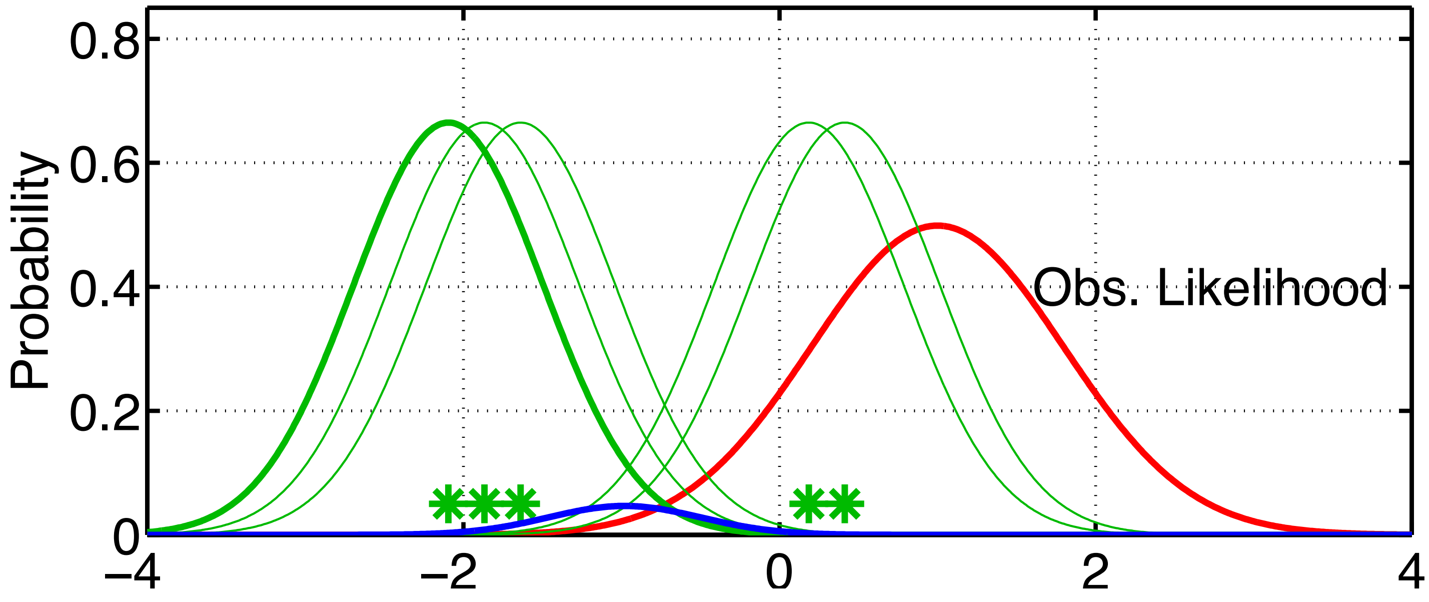
Ensemble Kernel Filter (EKF) (*filter_kind*=3 in *assim_tools_nml*).



Compute product of first kernel with Obs. Likelihood.

Ensemble Filter Algorithms:

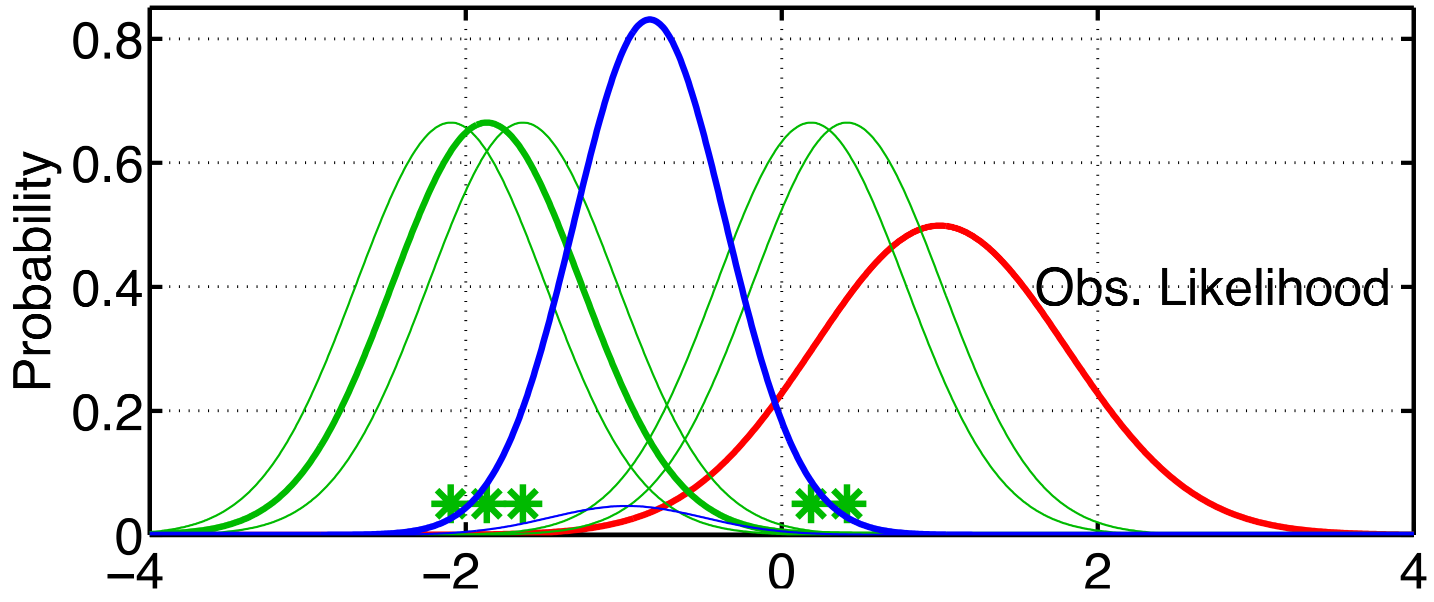
Ensemble Kernel Filter (EKF) (*filter_kind*=3 in *assim_tools_nml*).



But, can no longer ignore the weight term for product of Gaussians.
Kernels with mean further from observation get less weight.

Ensemble Filter Algorithms:

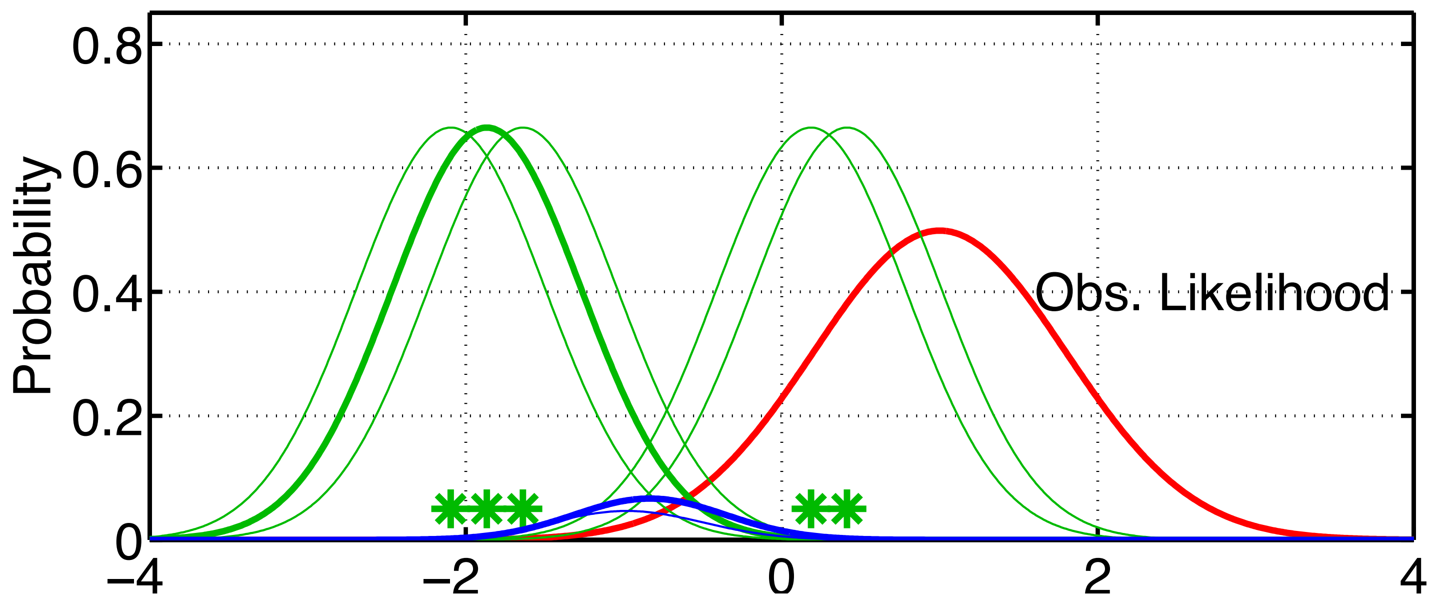
Ensemble Kernel Filter (EKF) (*filter_kind*=3 in *assim_tools_nml*).



Continue to take products for each kernel in turn.
More distant kernels have smaller impact on posterior.

Ensemble Filter Algorithms:

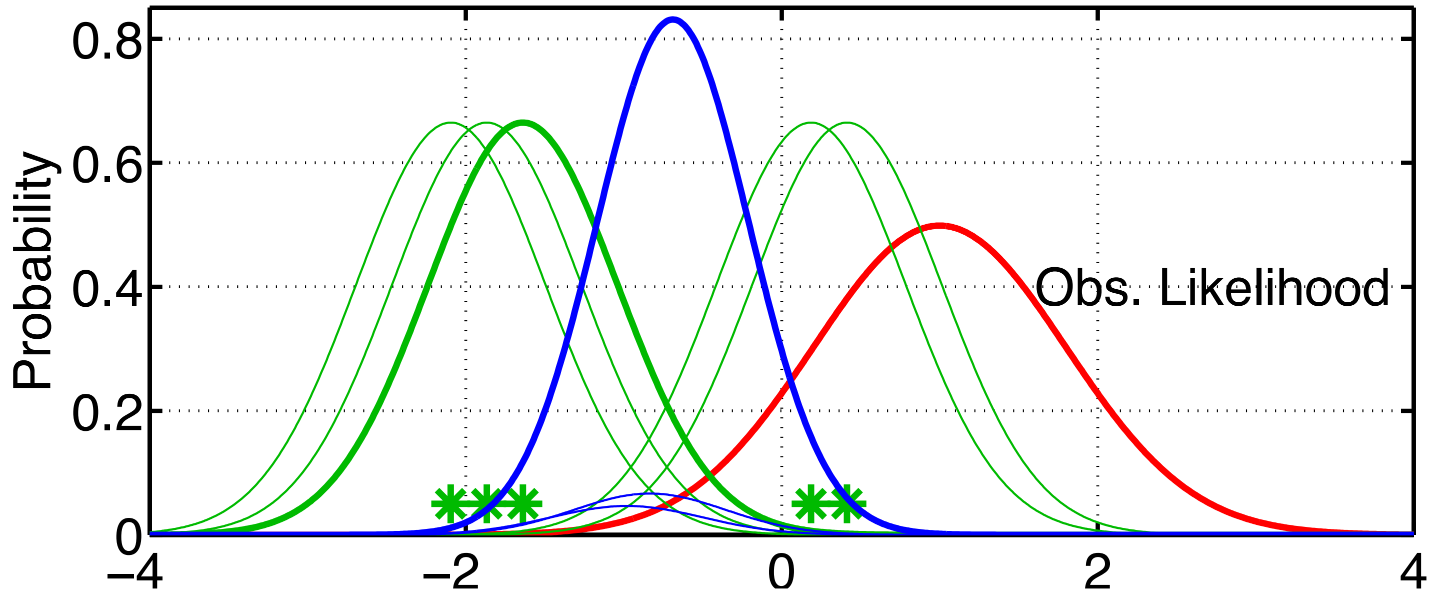
Ensemble Kernel Filter (EKF) (*filter_kind*=3 in *assim_tools_nml*).



Continue to take products for each kernel in turn.
More distant kernels have smaller impact on posterior.

Ensemble Filter Algorithms:

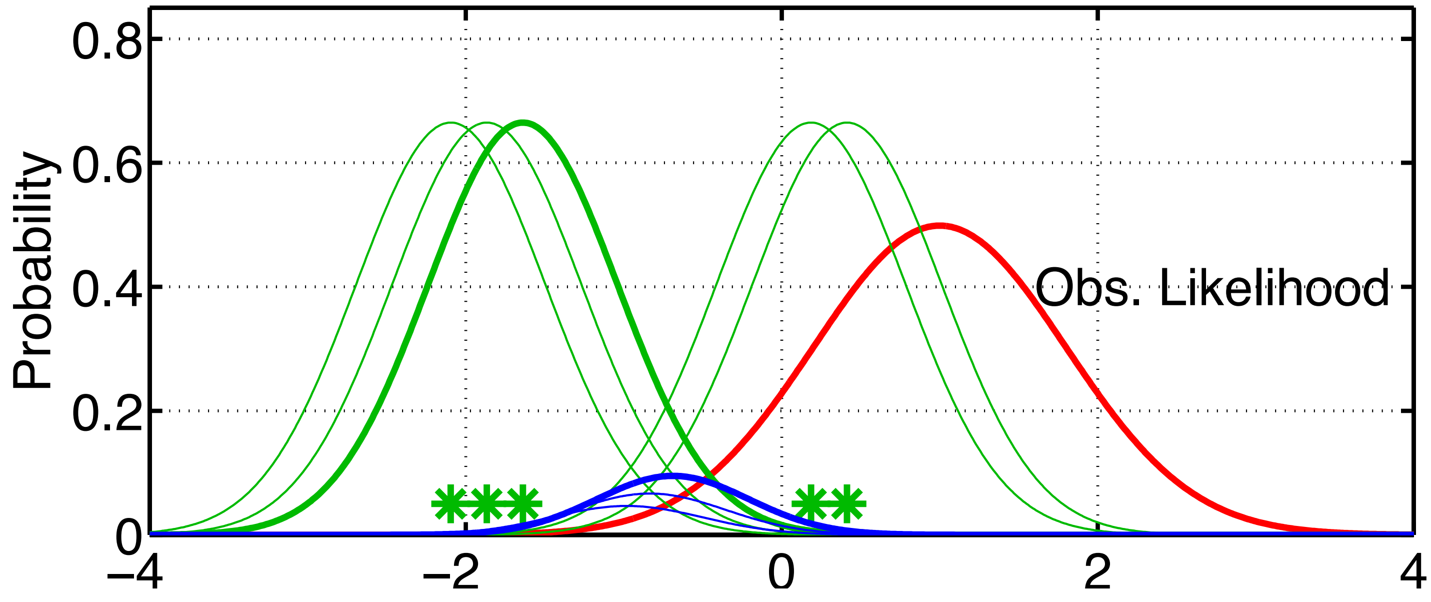
Ensemble Kernel Filter (EKF) (*filter_kind*=3 in *assim_tools_nml*).



Continue to take products for each kernel in turn.
More distant kernels have smaller impact on posterior.

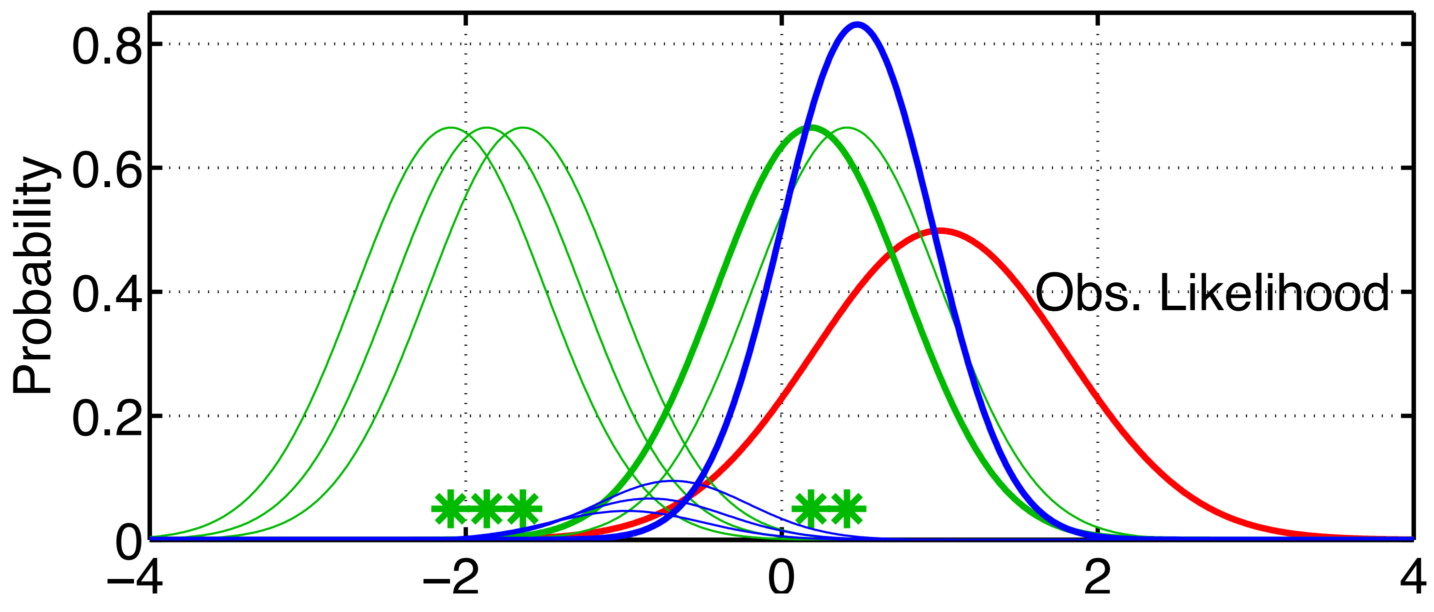
Ensemble Filter Algorithms:

Ensemble Kernel Filter (EKF) (*filter_kind*=3 in *assim_tools_nml*).



Ensemble Filter Algorithms:

Ensemble Kernel Filter (EKF) (*filter_kind*=3 in *assim_tools_nml*).

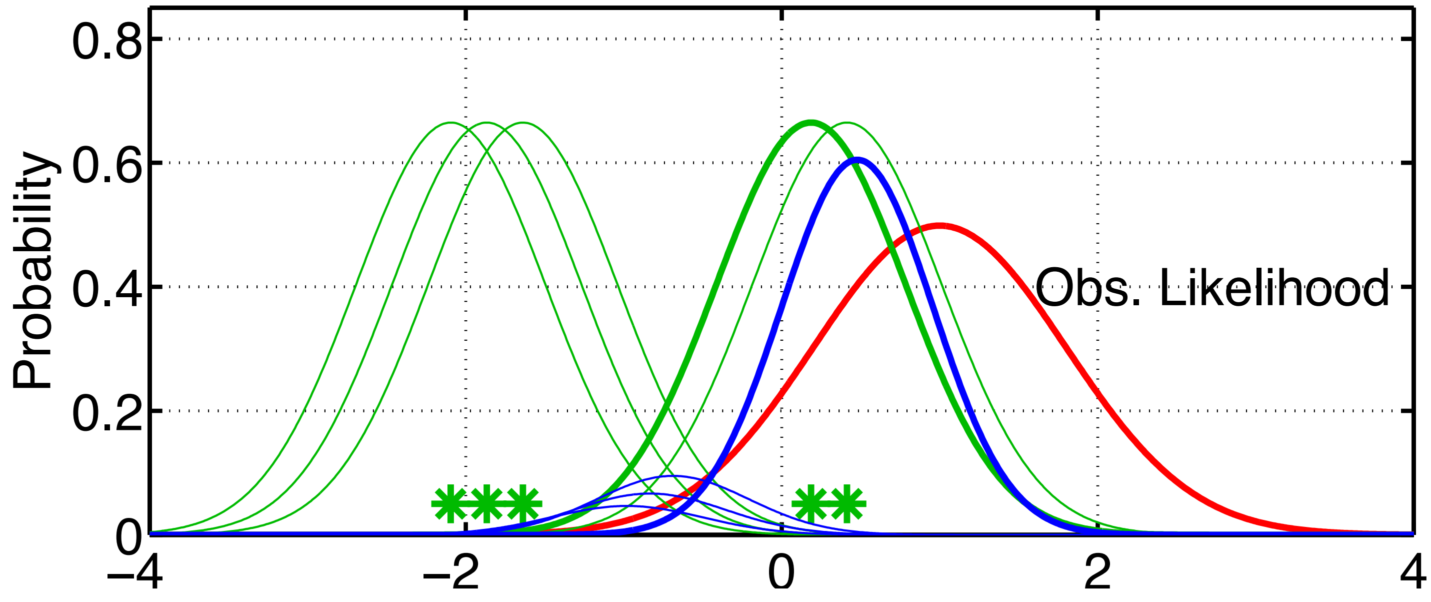


Continue to take products for each kernel in turn.

Closer kernels dominate posterior.

Ensemble Filter Algorithms:

Ensemble Kernel Filter (EKF) (*filter_kind*=3 in *assim_tools_nml*).

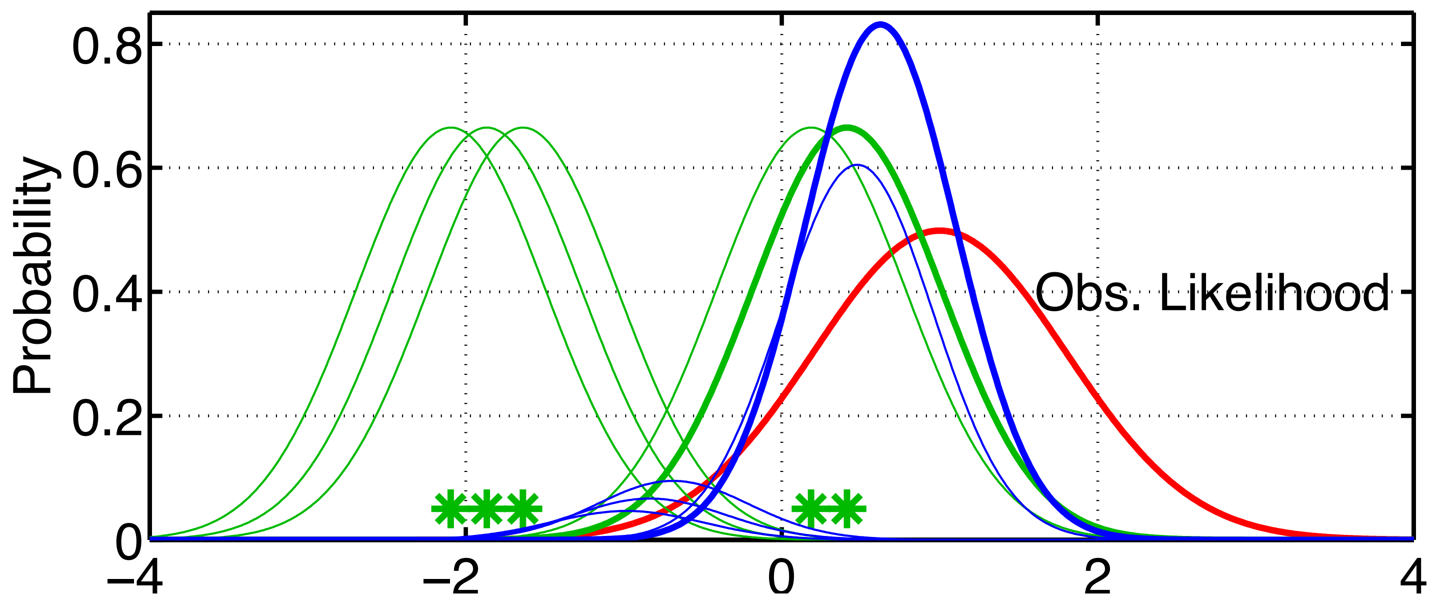


Continue to take products for each kernel in turn.

Closer kernels dominate posterior.

Ensemble Filter Algorithms:

Ensemble Kernel Filter (EKF) (*filter_kind*=3 in *assim_tools_nml*).

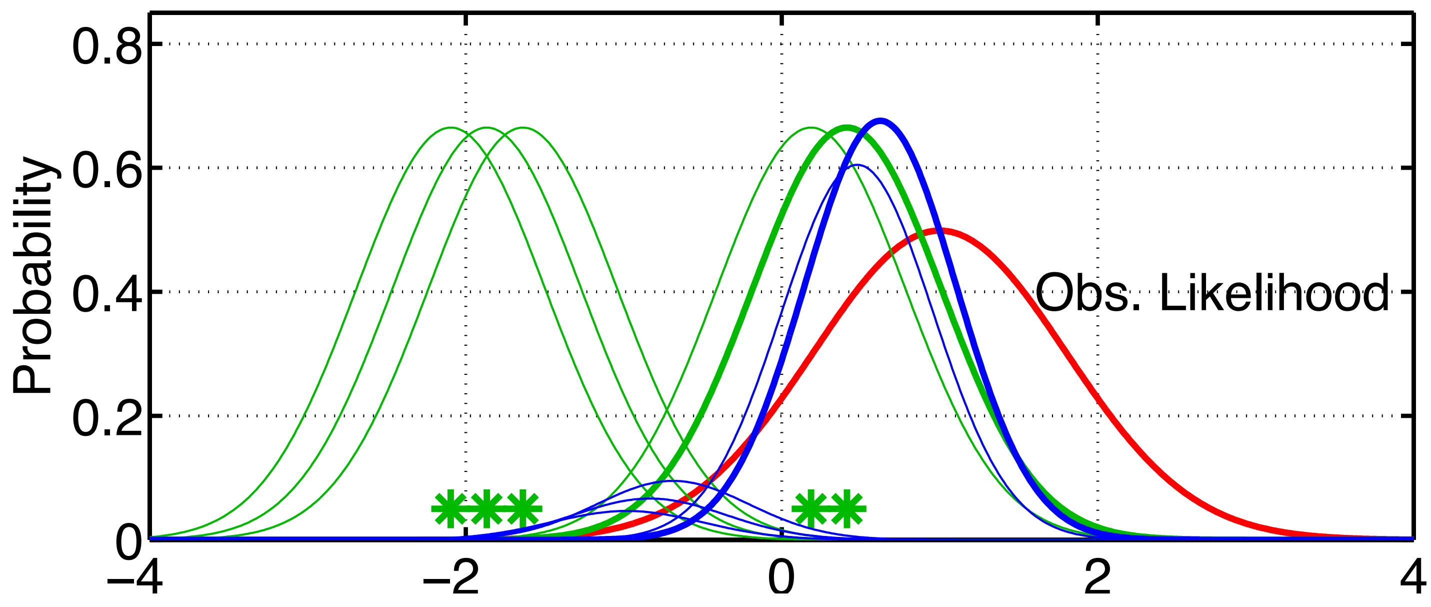


Continue to take products for each kernel in turn.

Closer kernels dominate posterior.

Ensemble Filter Algorithms:

Ensemble Kernel Filter (EKF) (*filter_kind*=3 in *assim_tools_nml*).

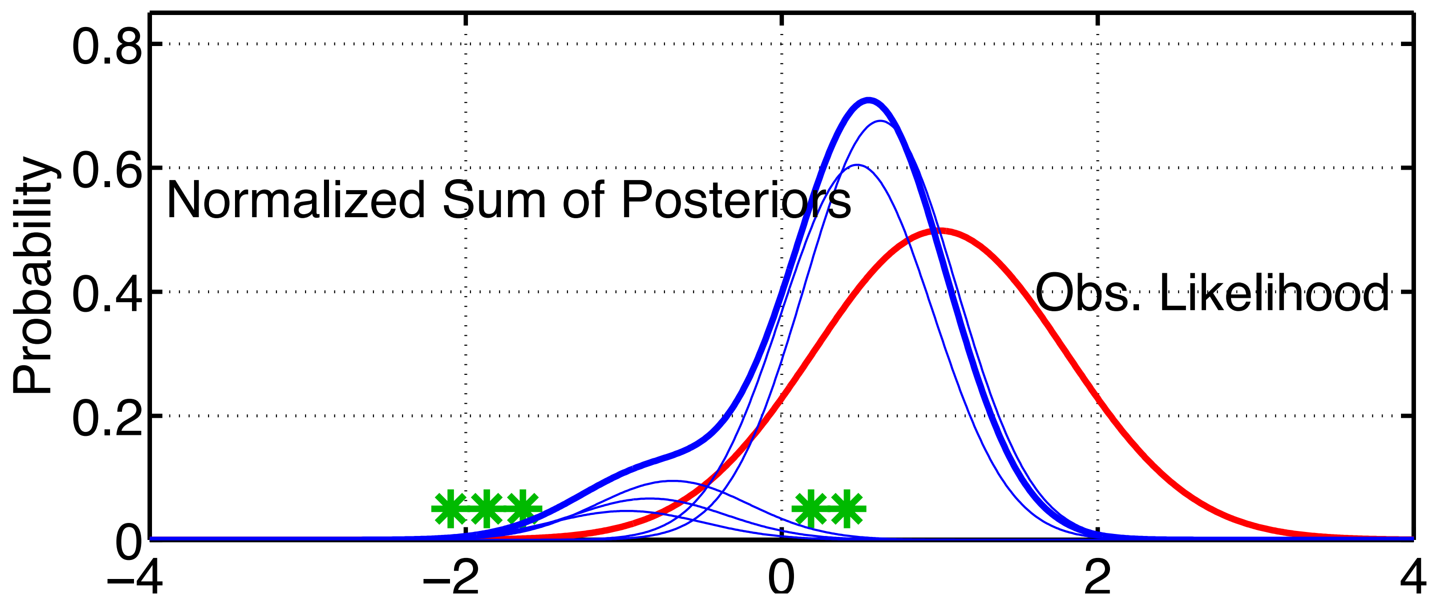


Continue to take products for each kernel in turn.

Closer kernels dominate posterior.

Ensemble Filter Algorithms:

Ensemble Kernel Filter (EKF) (*filter_kind*=3 in *assim_tools_nml*).

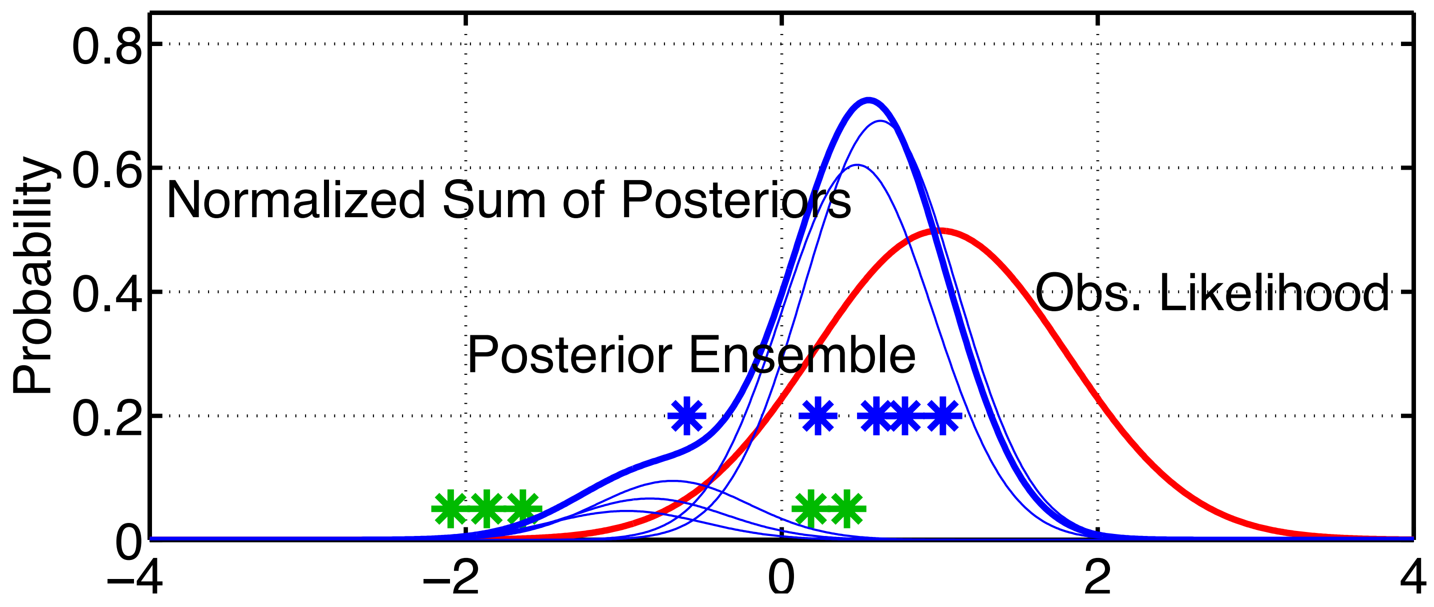


Final posterior is weight-normalized sum of kernel products.

Posterior is somewhat different than for ensemble adjustment or ensemble Kalman filter (much less density in left lobe.)

Ensemble Filter Algorithms:

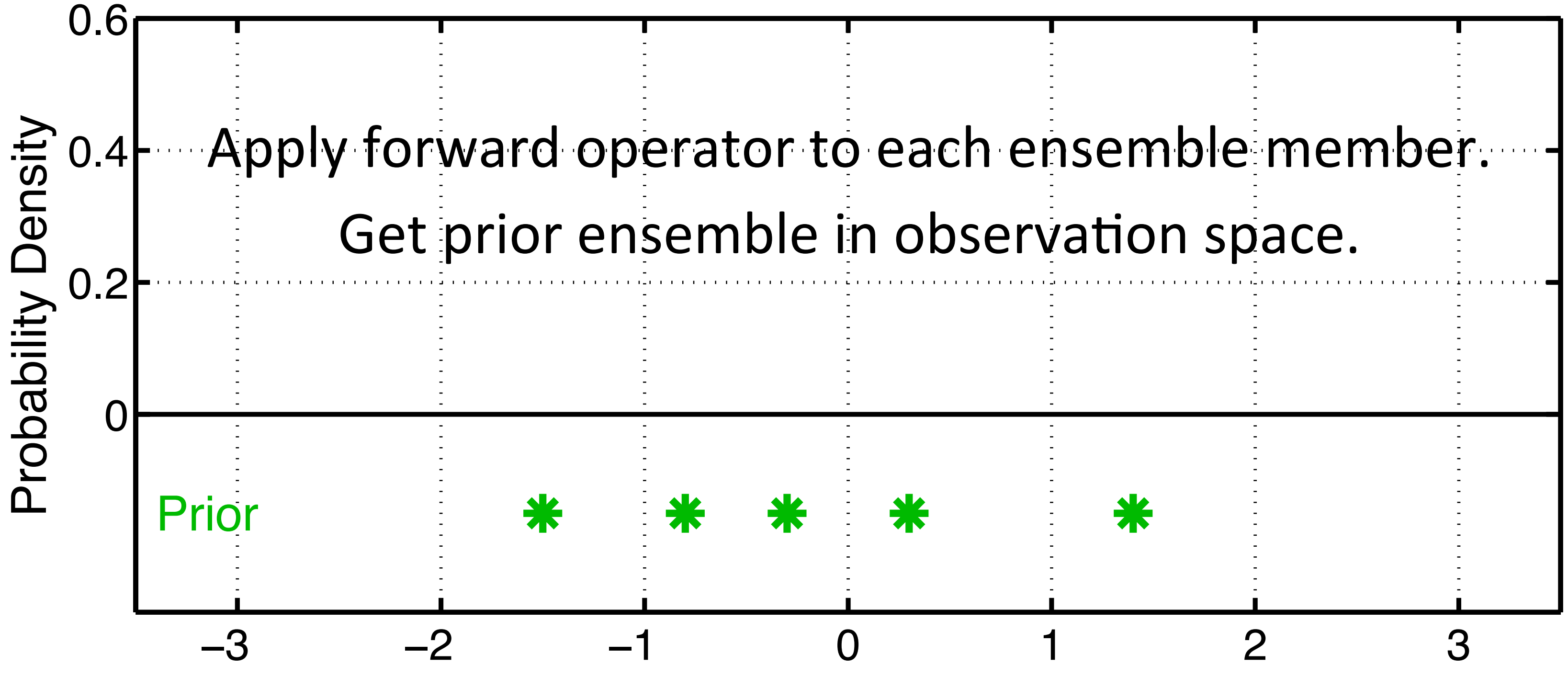
Ensemble Kernel Filter (EKF) (*filter_kind*=3 in *assim_tools_nml*).



Forming sample of the posterior can be problematic.
Random sample is simple.
Deterministic sampling is much more tricky here (few results available).

Ensemble Filter Algorithms:

Rank Histogram Filter (*filter_kind*=8 in *assim_tools_nml*).



Apply forward operator to each ensemble member.

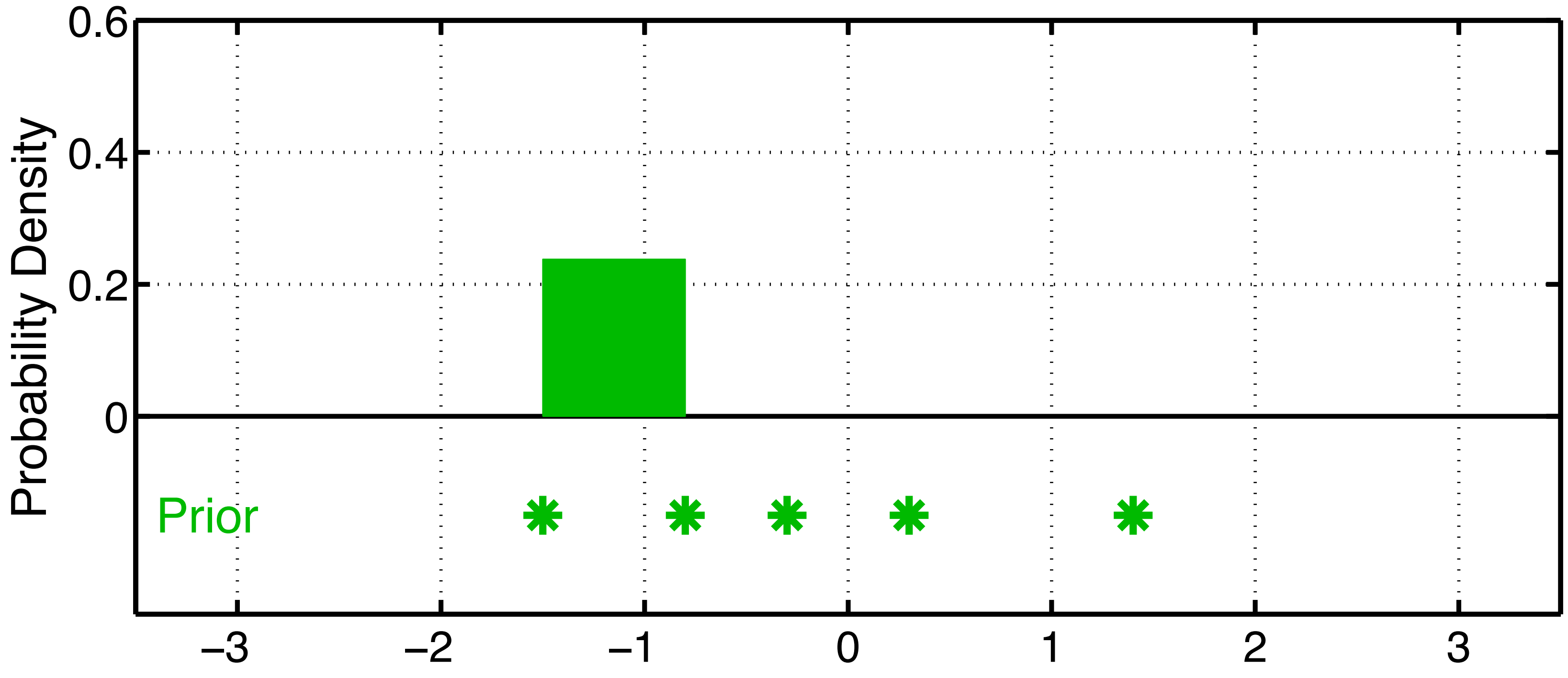
Get prior ensemble in observation space.

Goal: Want to handle non-Gaussian priors or observation likelihoods.
Low information content obs. must yield small increments.
Must perform well for Gaussian priors.
Must be computationally efficient.

Anderson, J. L., 2010: A Non-Gaussian Ensemble Filter Update for Data Assimilation.
Mon. Wea. Rev., **139**, 4186-4198. doi: 10.1175/2010MWR3253.1

Ensemble Filter Algorithms:

Rank Histogram Filter (*filter_kind*=8 in *assim_tools_nml*).

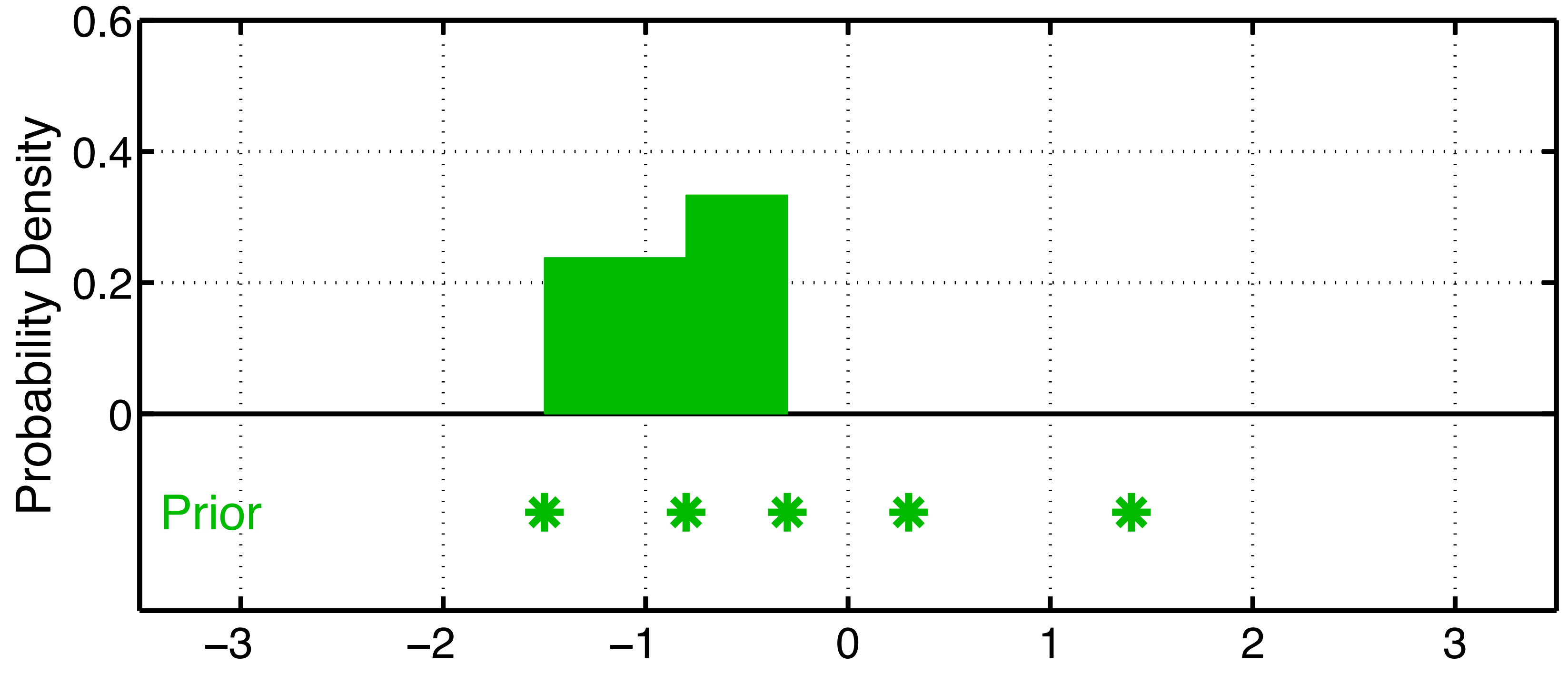


Step 1: Get continuous prior distribution density.

- Place (ens_size + 1)$^{-1}$ mass between adjacent ensemble members.
- Reminiscent of rank histogram evaluation method.

Ensemble Filter Algorithms:

Rank Histogram Filter (*filter_kind*=8 in *assim_tools_nml*).

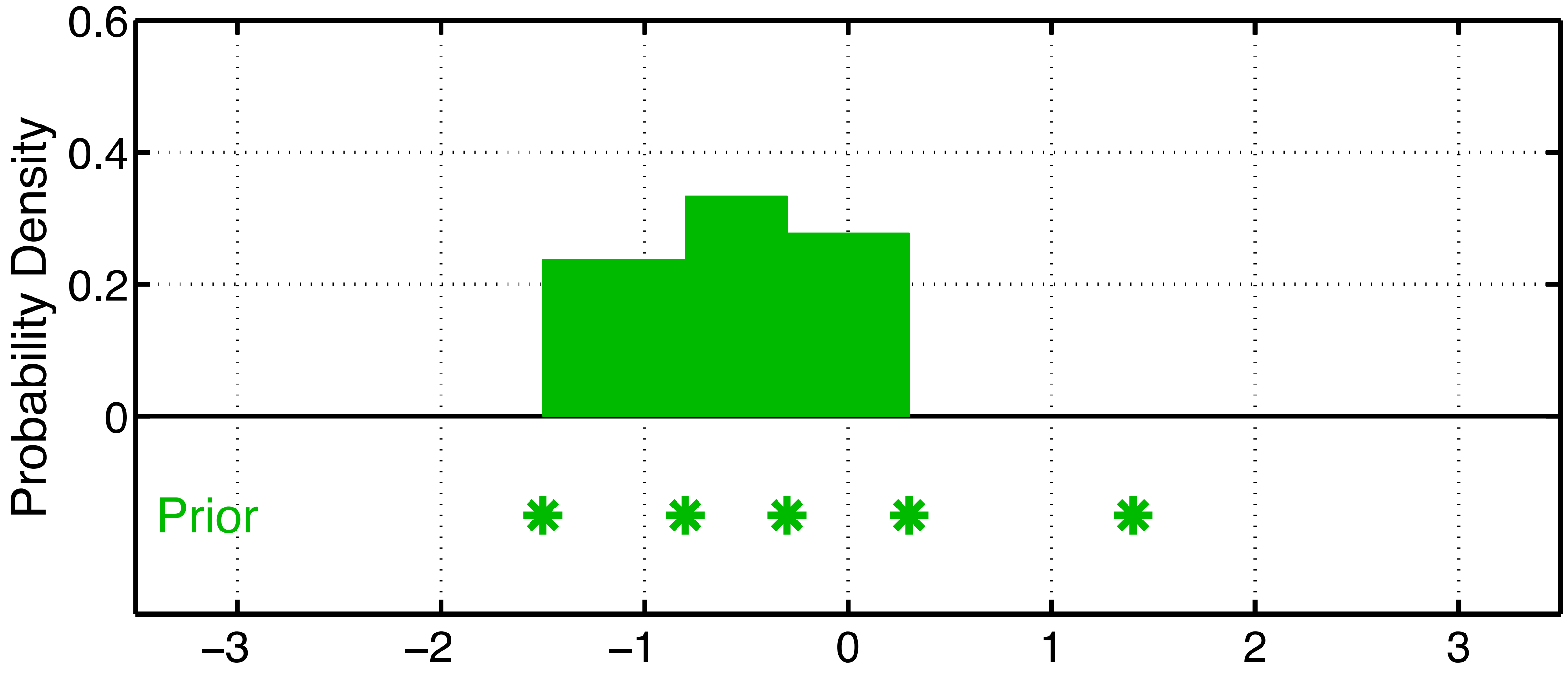


Step 1: Get continuous prior distribution density.

- Place (ens_size + 1)$^{-1}$ mass between adjacent ensemble members.
- Reminiscent of rank histogram evaluation method.

Ensemble Filter Algorithms:

Rank Histogram Filter (*filter_kind*=8 in *assim_tools_nml*).

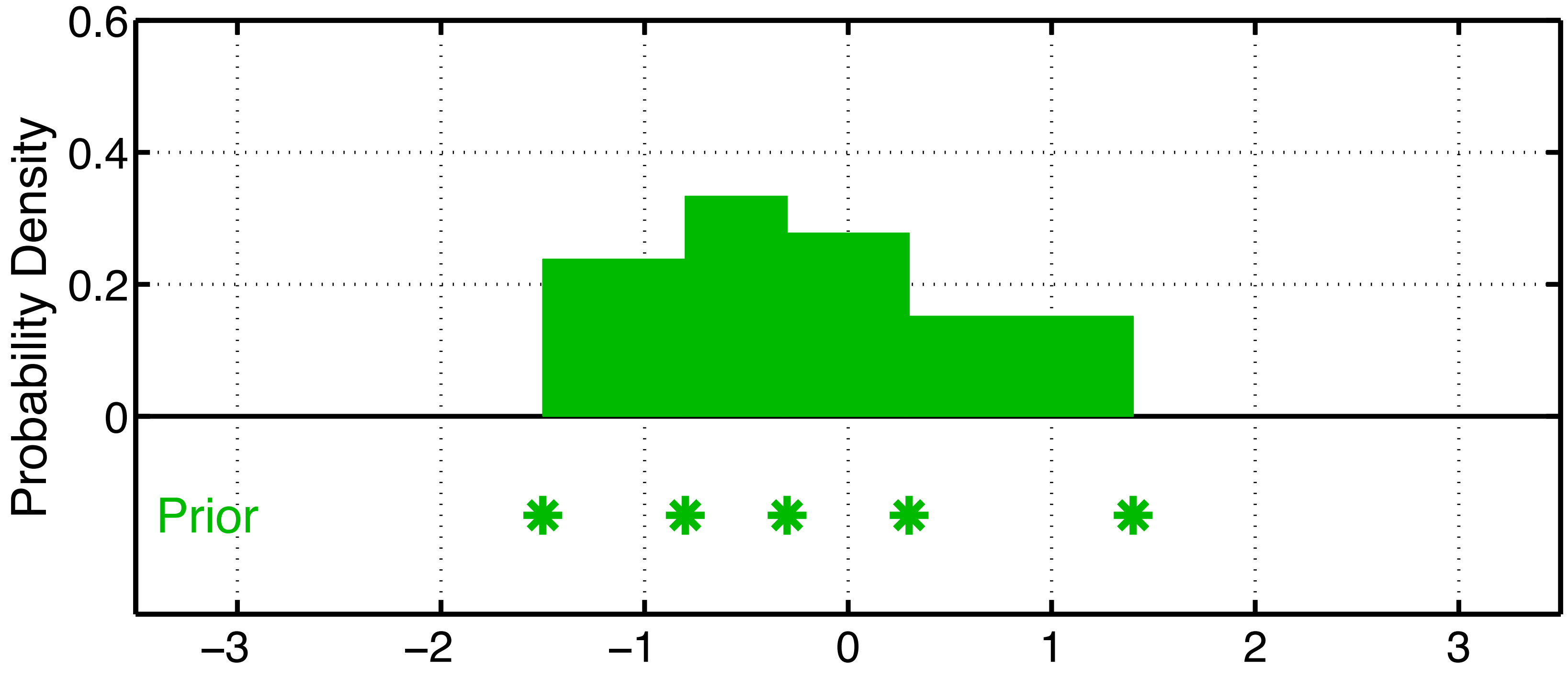


Step 1: Get continuous prior distribution density.

- Place (ens_size + 1)$^{-1}$ mass between adjacent ensemble members.
- Reminiscent of rank histogram evaluation method.

Ensemble Filter Algorithms:

Rank Histogram Filter (*filter_kind*=8 in *assim_tools_nml*).

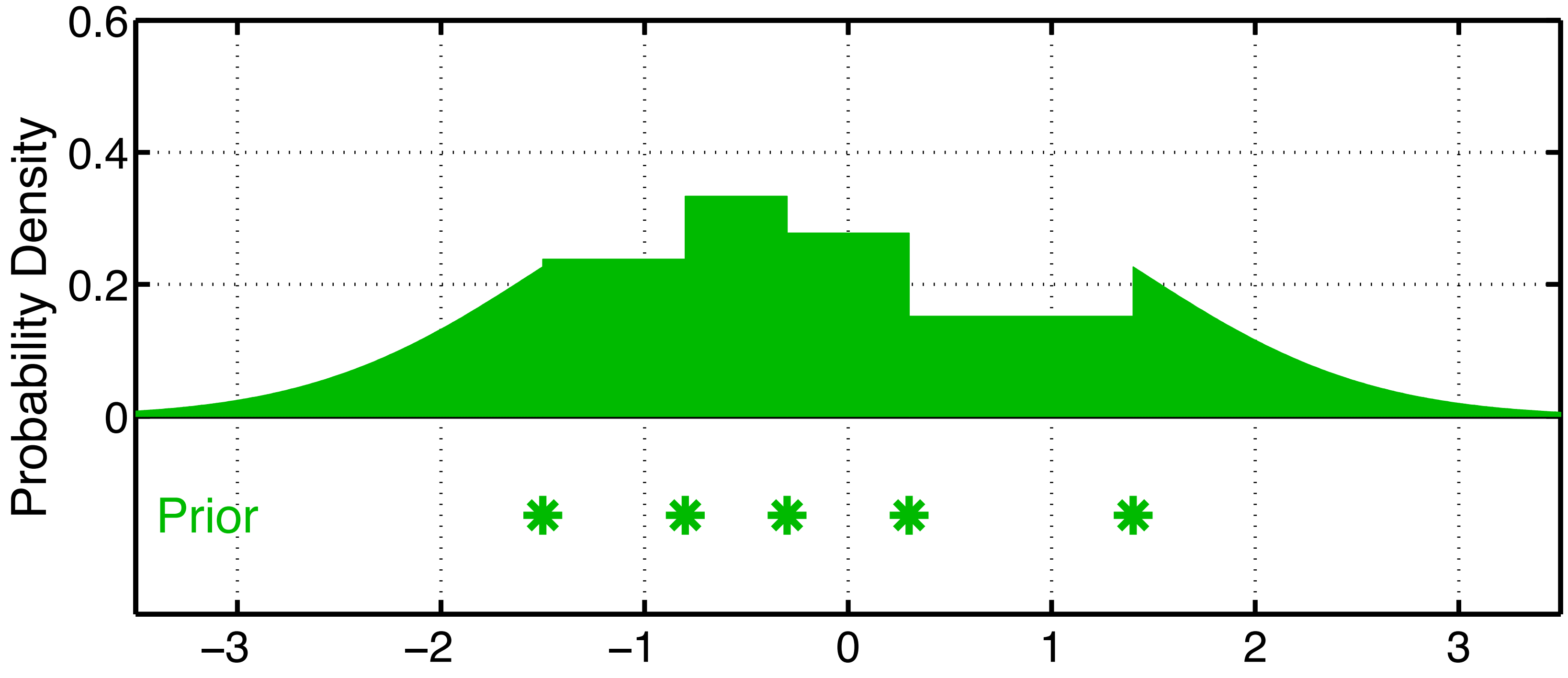


Step 1: Get continuous prior distribution density.

- Place (ens_size + 1)$^{-1}$ mass between adjacent ensemble members.
- Reminiscent of rank histogram evaluation method.

Ensemble Filter Algorithms:

Rank Histogram Filter (*filter_kind*=8 in *assim_tools_nml*).

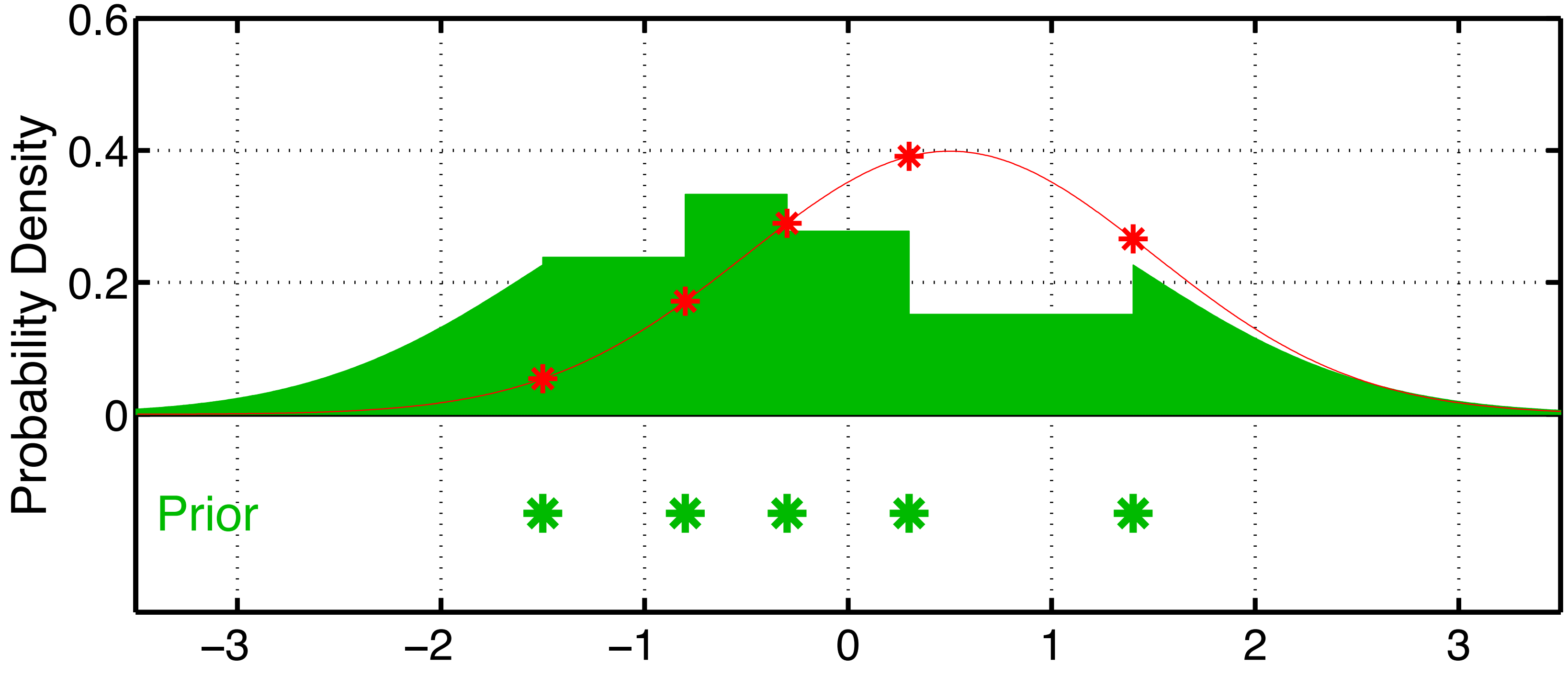


Step 1: Get continuous prior distribution density.

- Partial gaussian kernels on tails, N(*tail_mean*, *ens_sd*).
- *tail_mean selected so that (ens_size + 1)*$^{-1}$ *mass is in tail*.

Ensemble Filter Algorithms:

Rank Histogram Filter (*filter_kind*=8 in *assim_tools_nml*).

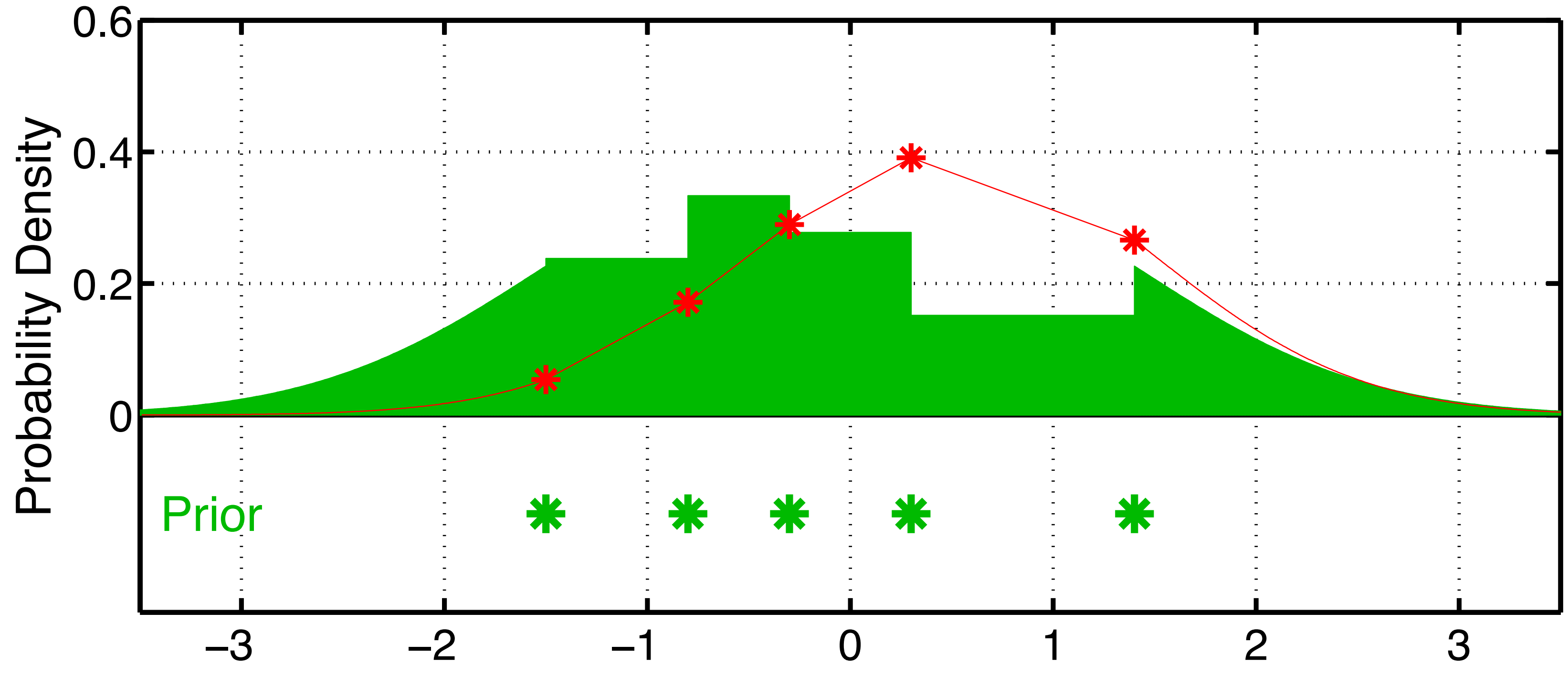


Step 2: Use likelihood to compute weight for each ensemble member.

- Analogous to classical particle filter.
- Can be extended to non-gaussian obs. likelihoods.

Ensemble Filter Algorithms:

Rank Histogram Filter (*filter_kind*=8 in *assim_tools_nml*).

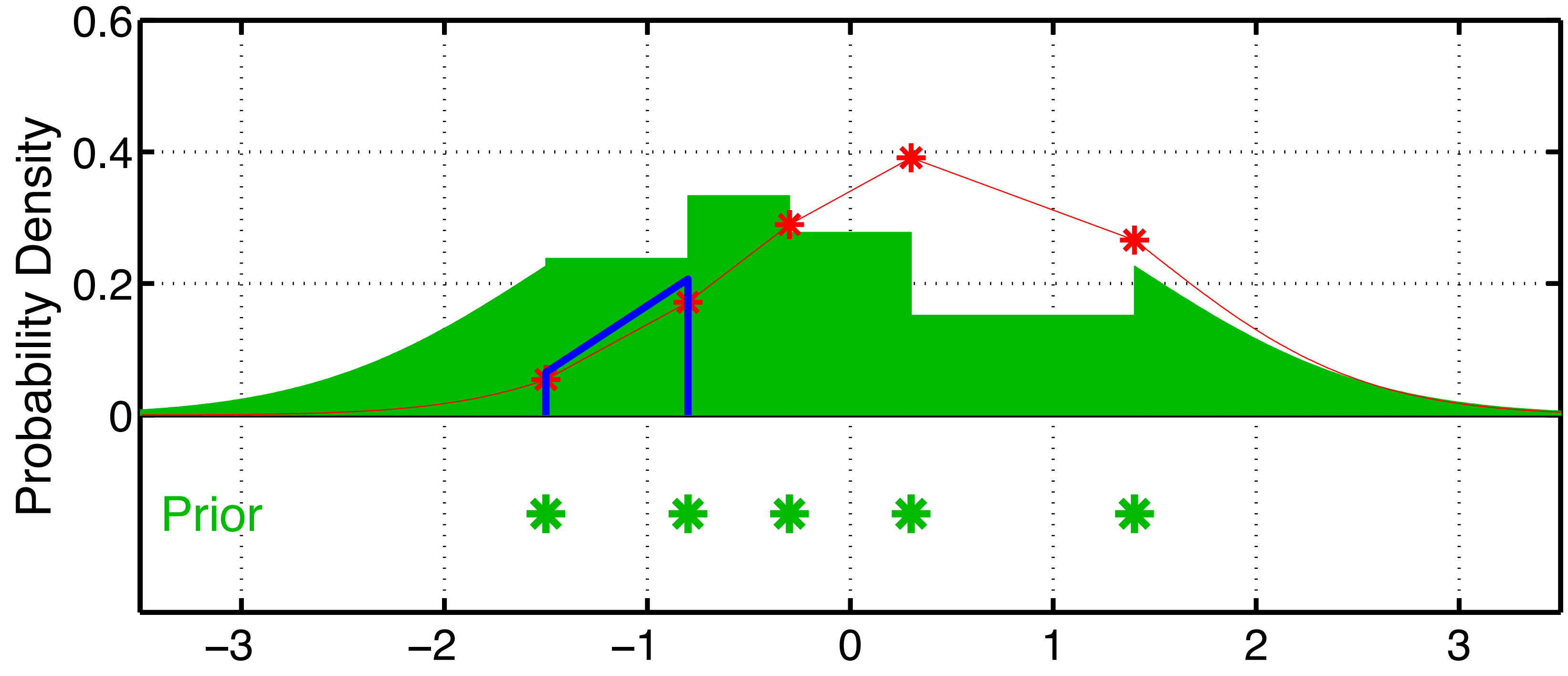


Step 2: Use likelihood to compute weight for each ensemble member.

- Can approximate interior likelihood with linear fit; for efficiency.

Ensemble Filter Algorithms:

Rank Histogram Filter (*filter_kind*=8 in *assim_tools_nml*).



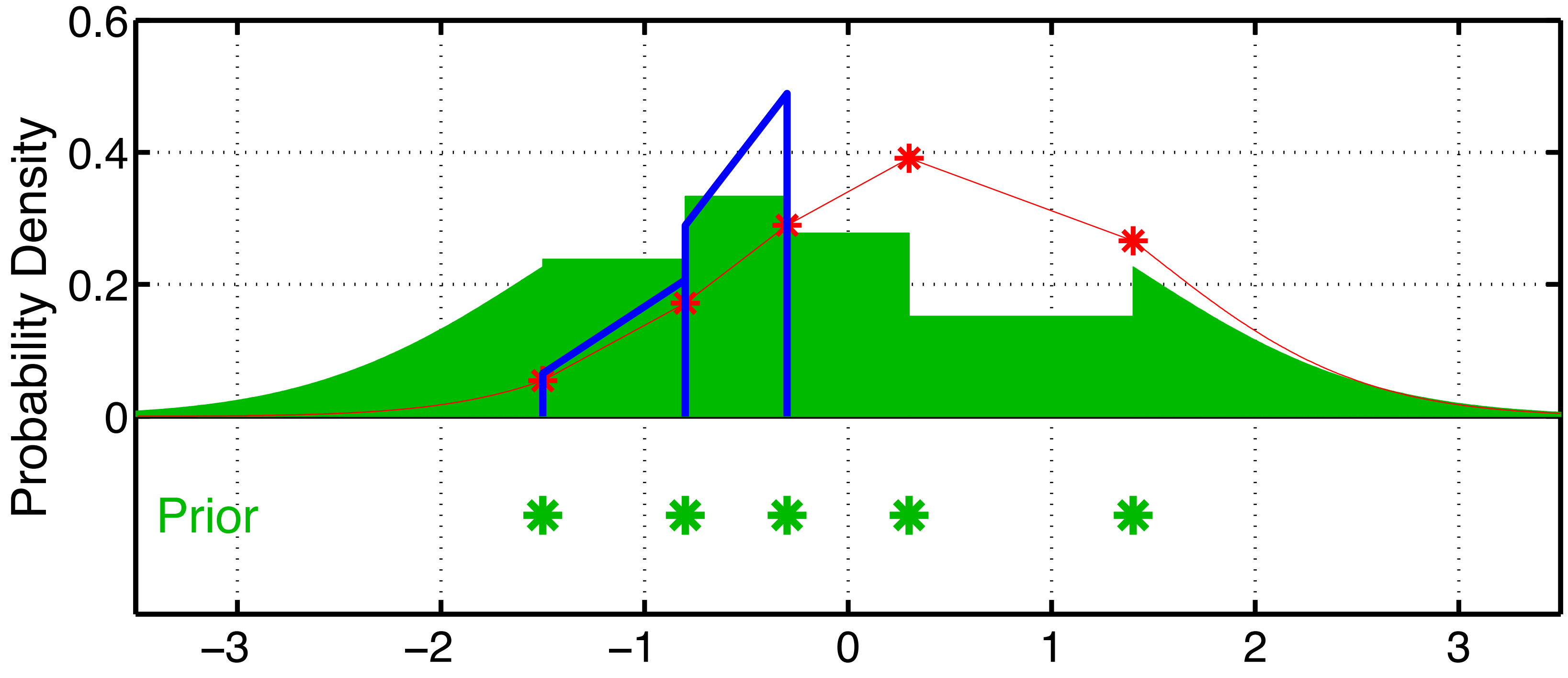
Step 3: Compute continuous posterior distribution.

- Approximate likelihood with trapezoidal quadrature, take product.

  (Displayed product normalized to make posterior a PDF).

Ensemble Filter Algorithms:

Rank Histogram Filter (*filter_kind*=8 in *assim_tools_nml*).



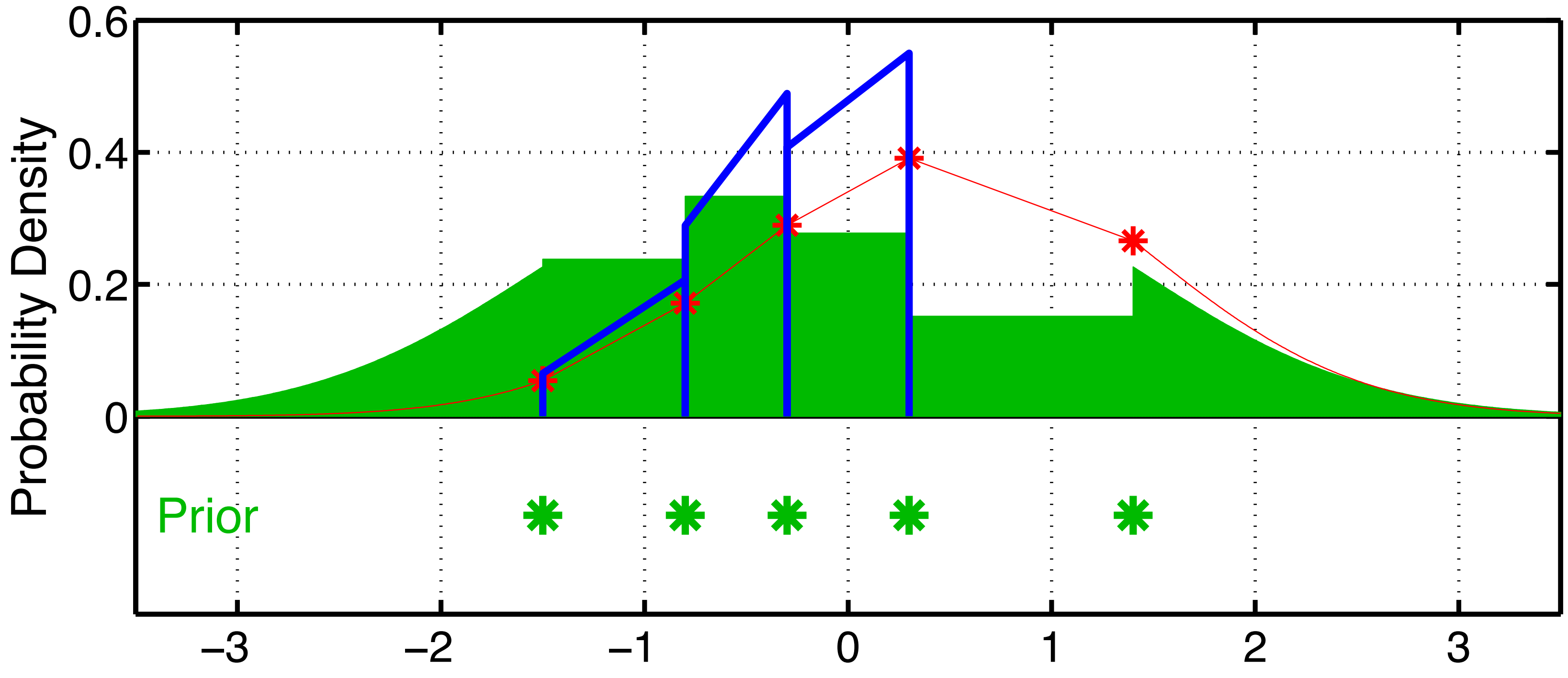
Step 3: Compute continuous posterior distribution.

- Approximate likelihood with trapezoidal quadrature, take product.

  (Displayed product normalized to make posterior a PDF).

Ensemble Filter Algorithms:

Rank Histogram Filter (*filter_kind*=8 in *assim_tools_nml*).

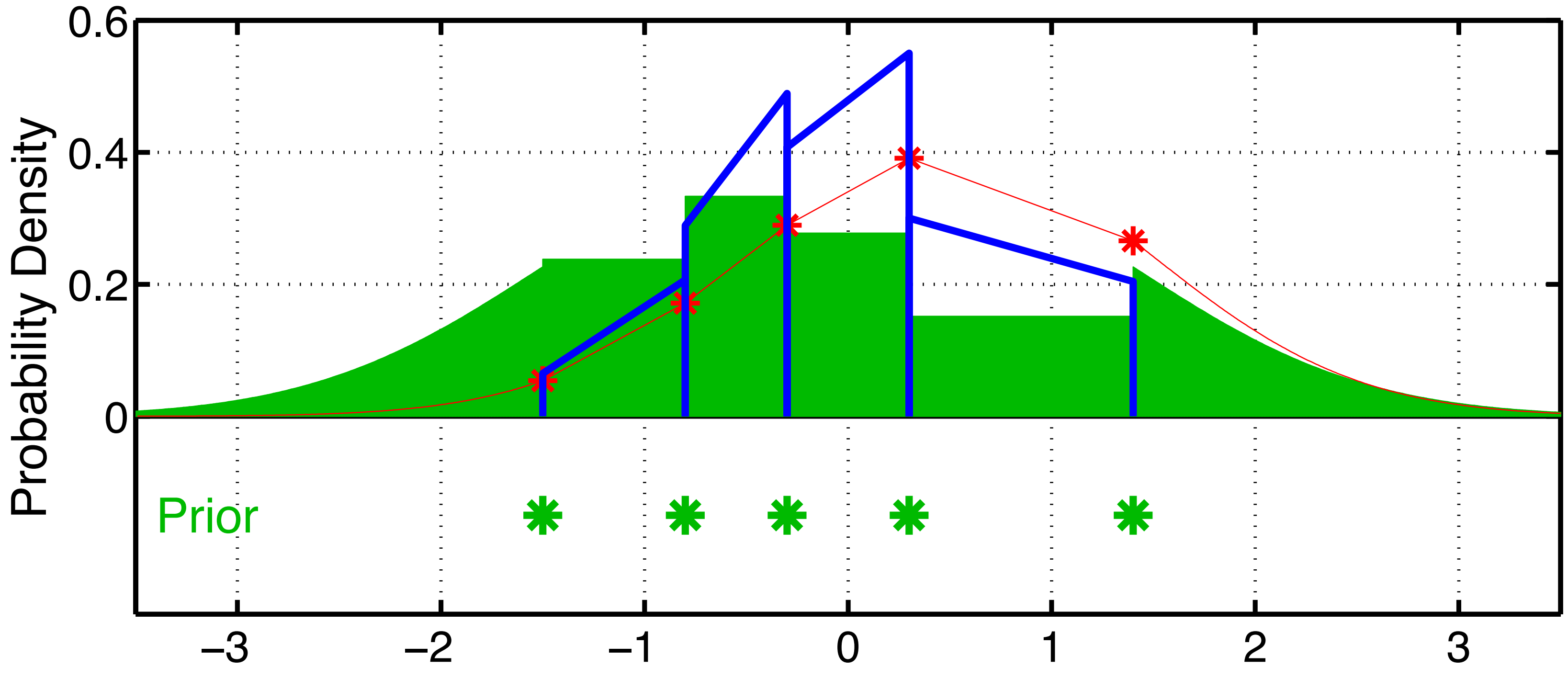


Step 3: Compute continuous posterior distribution.

- Approximate likelihood with trapezoidal quadrature, take product.
  (Displayed product normalized to make posterior a PDF).

Ensemble Filter Algorithms:

Rank Histogram Filter (*filter_kind*=8 in *assim_tools_nml*).

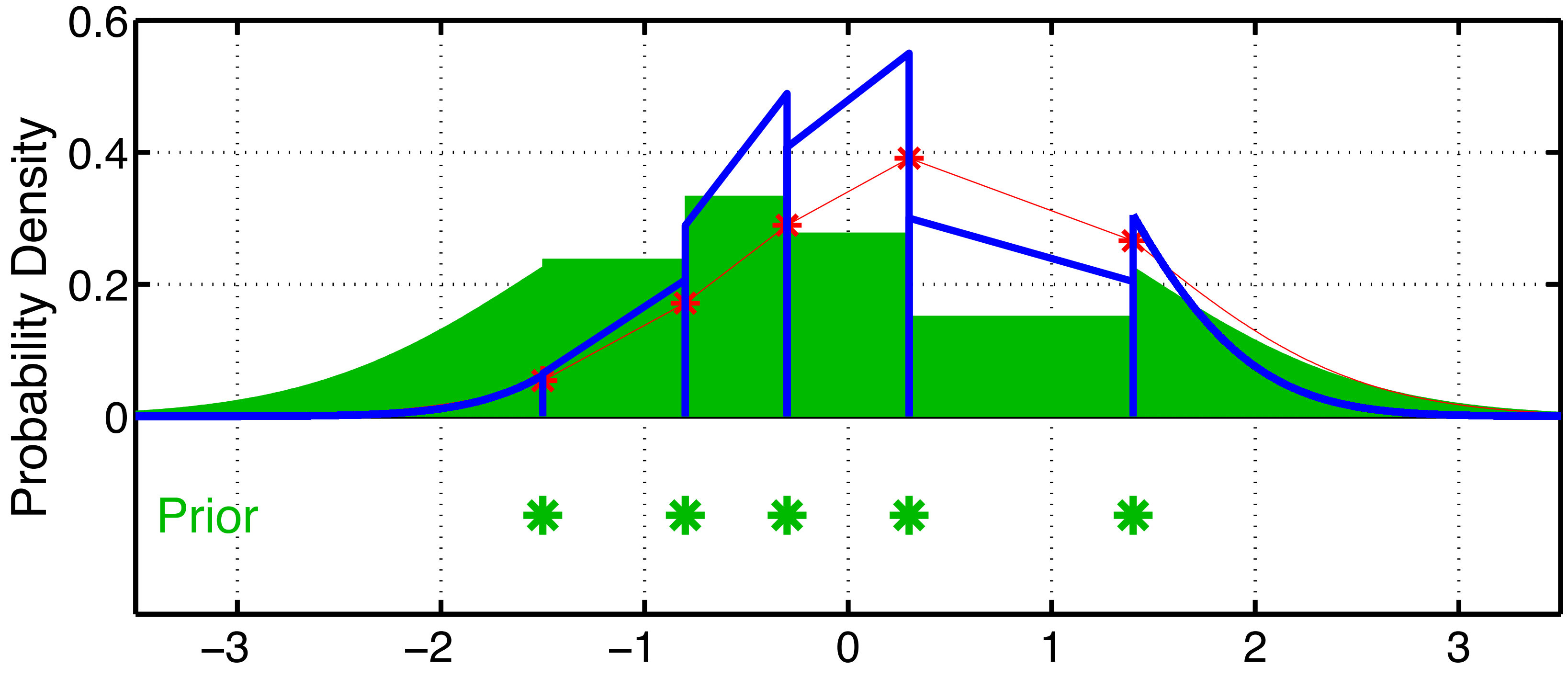


Step 3: Compute continuous posterior distribution.

- Approximate likelihood with trapezoidal quadrature, take product.
  (Displayed product normalized to make posterior a PDF).

Ensemble Filter Algorithms:

Rank Histogram Filter (*filter_kind*=8 in *assim_tools_nml*).

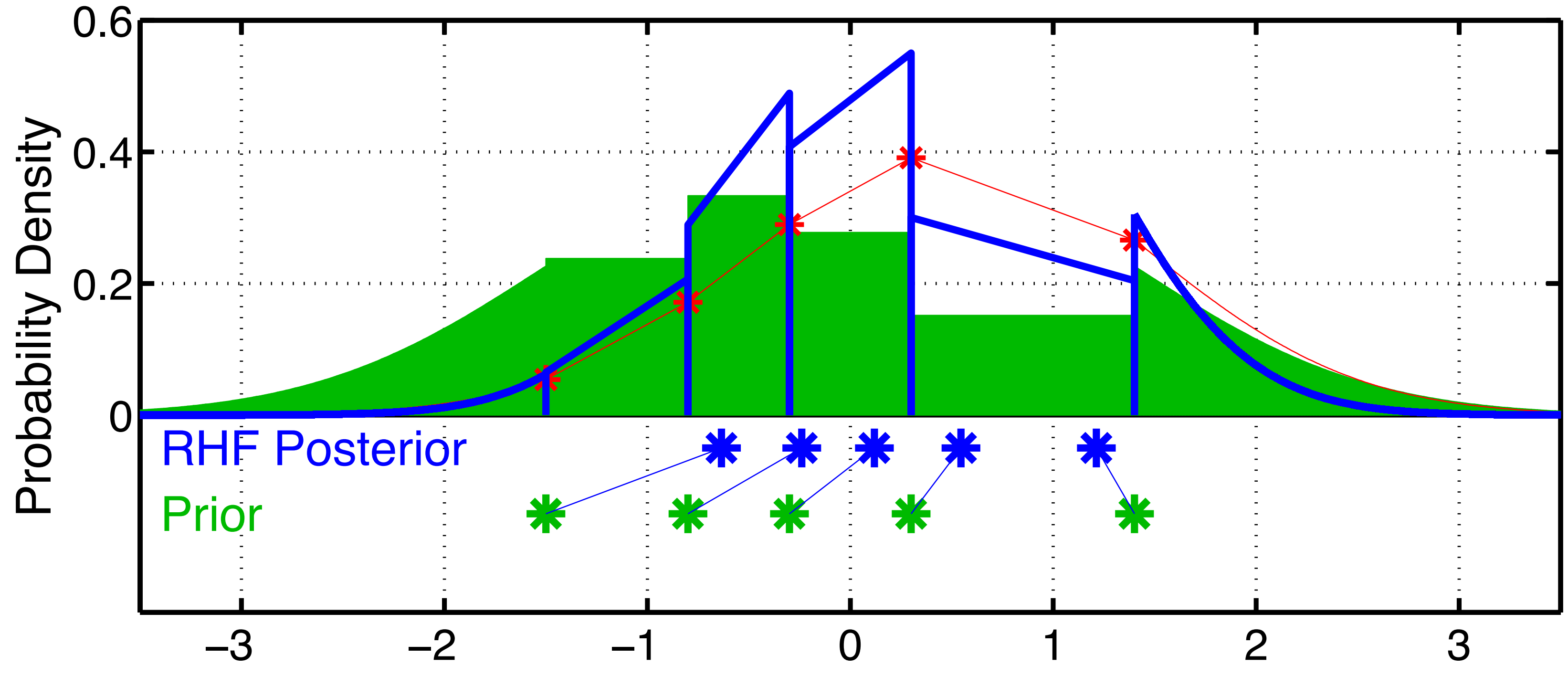


Step 3: Compute continuous posterior distribution.

- Product of prior gaussian kernel with likelihood for tails.
- Easy for gaussian likelihood.
- More quadrature if non-Gaussian likelihood.

Ensemble Filter Algorithms:

Rank Histogram Filter (*filter_kind*=8 in *assim_tools_nml*).

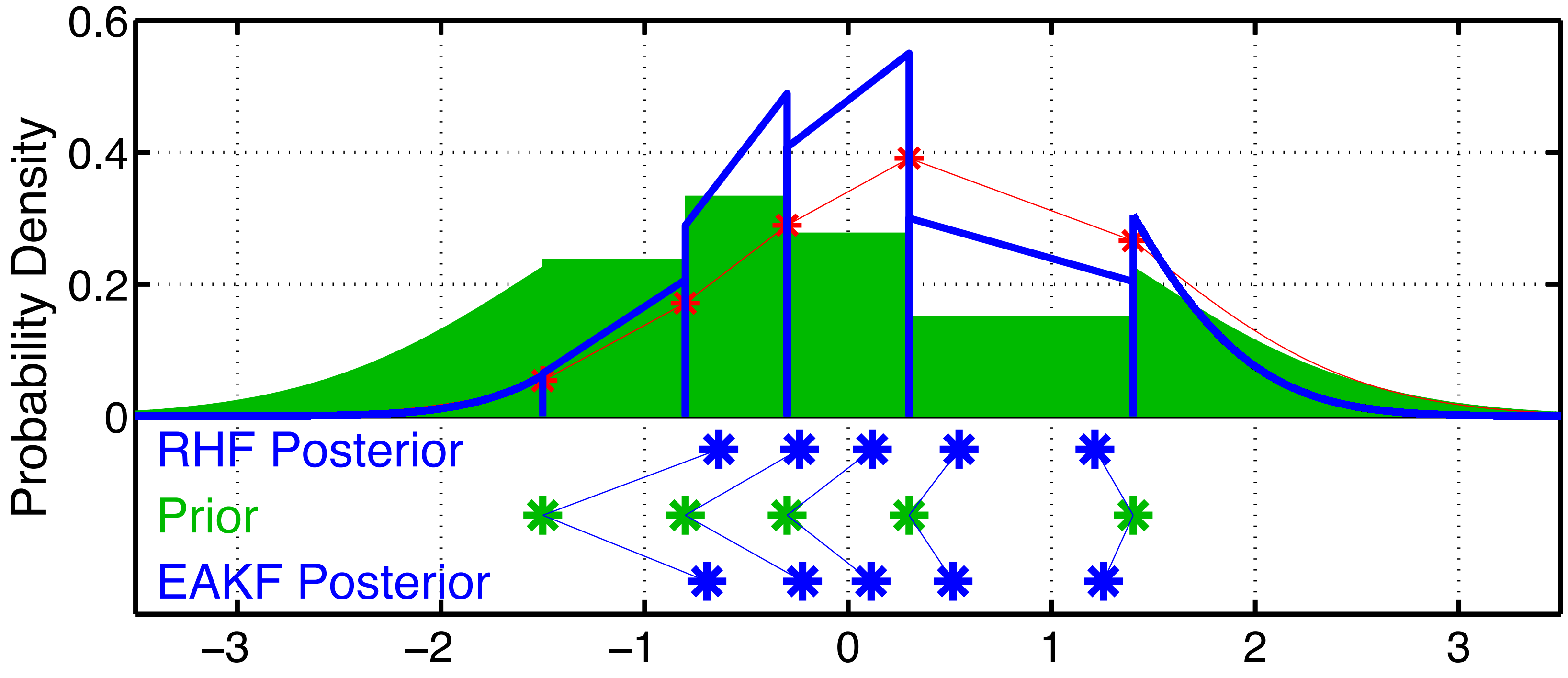


Step 4: Compute updated ensemble members:

- (ens_size +1)$^{-1}$ of posterior mass between each ensemble pair.
- (ens_size +1)$^{-1}$ in each tail.
- Uninformative observation has no impact.

Ensemble Filter Algorithms:

Rank Histogram Filter (*filter_kind*=8 in *assim_tools_nml*).

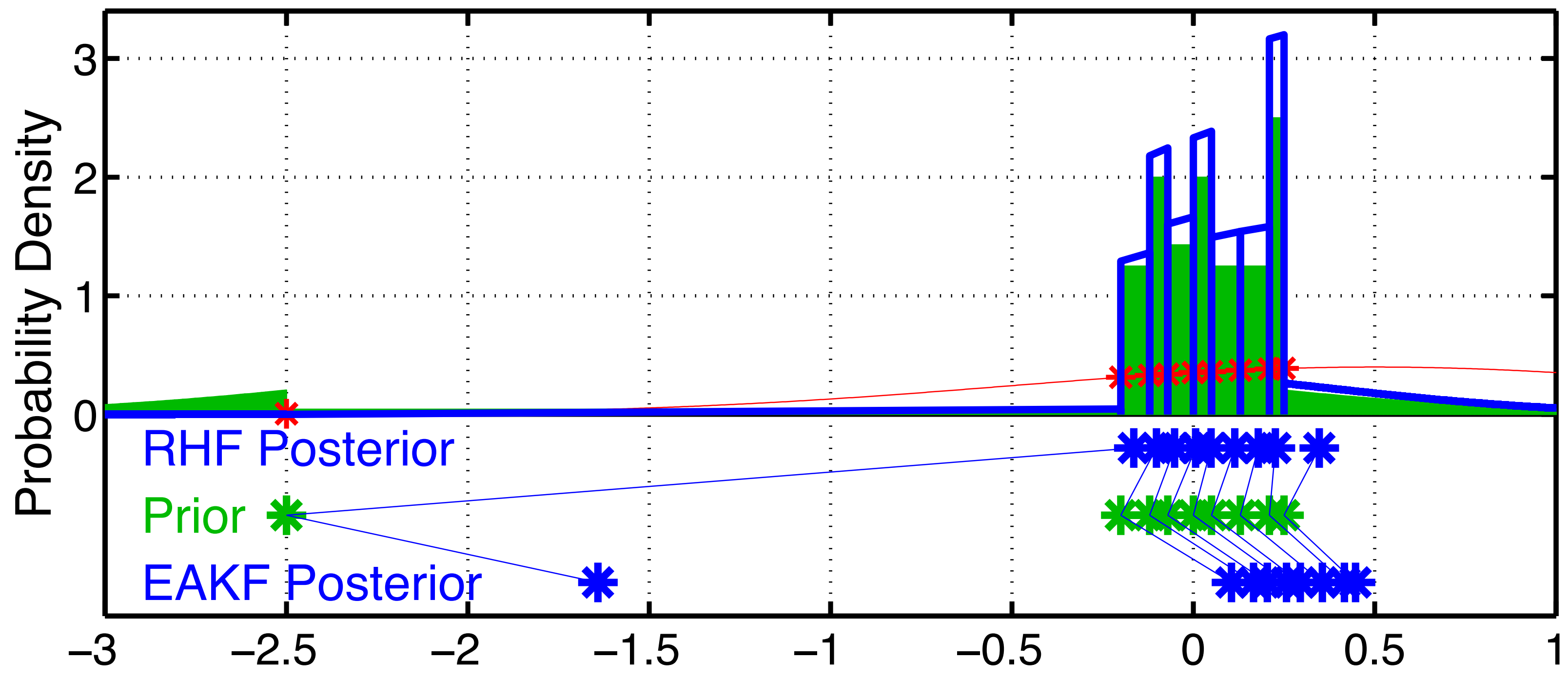


Compare to standard Ensemble Adjustment Filter (EAKF).

Nearly gaussian case, differences are small.

Ensemble Filter Algorithms:

Rank Histogram Filter (*filter_kind*=8 in *assim_tools_nml*).



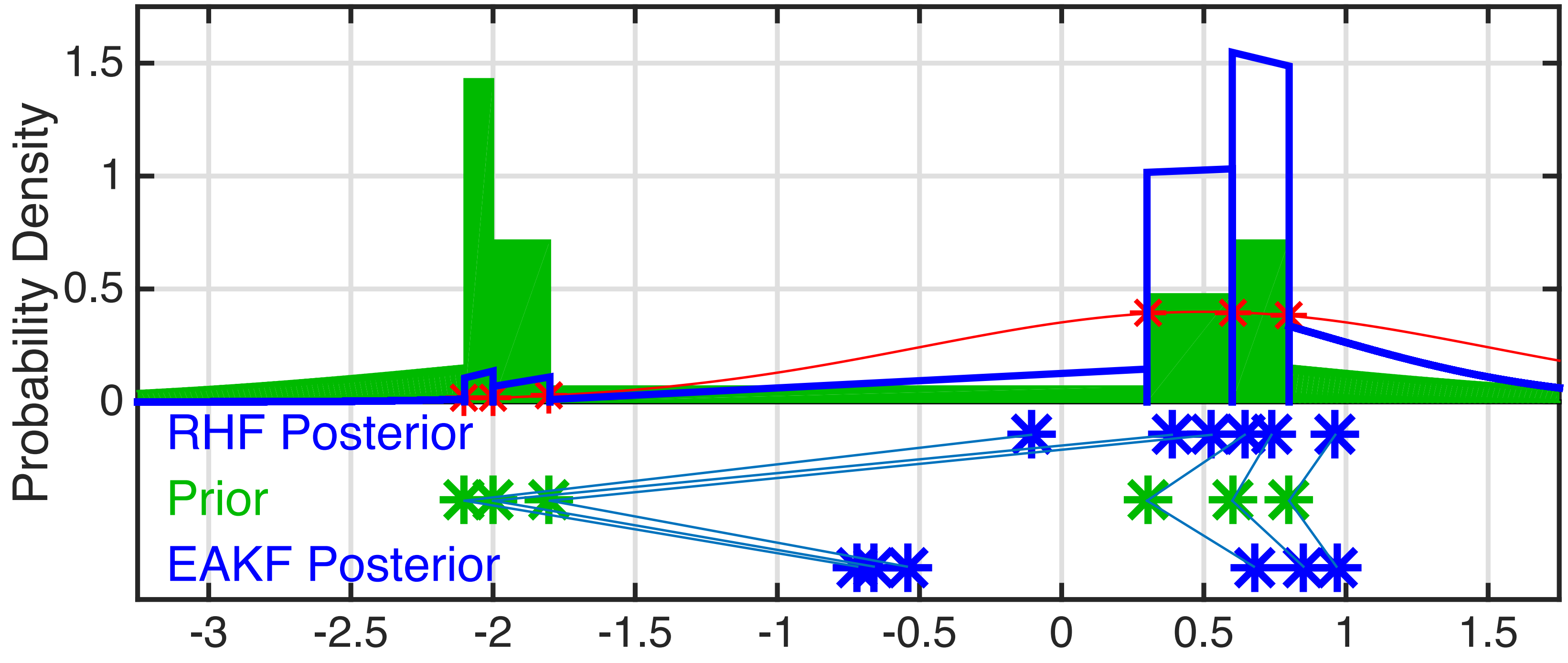
Rank Histogram gets rid of outlier that is clearly inconsistent with obs.

EAKF can't get rid of outlier.

Large prior variance from outlier causes EAKF to shift all members too much towards observation.

Ensemble Filter Algorithms:

Rank Histogram Filter (*filter_kind*=8 in *assim_tools_nml*).



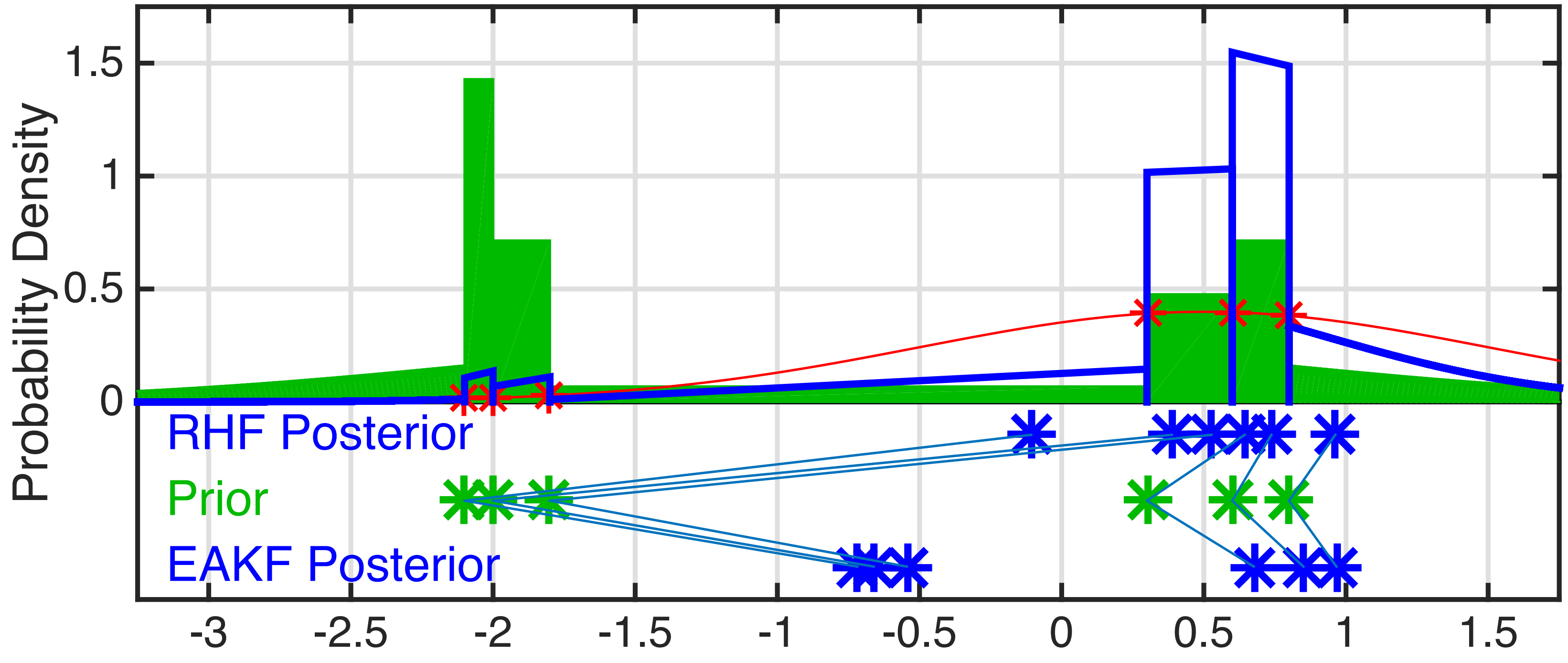
Rank Histogram handles multimodal prior and compelling observation.

EAKF still bimodal; left mode is inconsistent with everything.

Lorenz_63 can have priors like this.

Ensemble Filter Algorithms:

Rank Histogram Filter (*filter_kind*=8 in *assim_tools_nml*).



Convective-scale models (and land models) have analogous behavior.

Convection may fire at 'random' locations.

Subset of ensembles will be in right place, rest in wrong place.

Want to aggressively eliminate convection in wrong place.

Ensemble Filter Algorithms:

Particle filter methods:

These are ‘classical’ ensemble methods from statistical literature.

Size of ensembles required scales hyper-exponentially with model size.

Ensembles > 1000 required for models with > 4 degrees of freedom.

This rules out naive application to any meaningful atmospheric model.

At present, nobody knows ways to attack this so no details here.

Ensemble Filter Algorithms:

Particle Filter (*filter_kind*=4 in *assim_tools_nml*).

Can use particle filters in a few dimensions.

DART provides a one-dimensional particle filter.

Independent particle filter is used for updating each observation.

PROBLEM: Inconsistency between updates for different observations.

This can probably be made to work in some clever way!

Using DART Diagnostics

What happens when these different methods are used in Lorenz_63?

Are they significantly different?

Do some work better for different observation sets?

Can kernel filter deal better with distinct bimodality of Lorenz_63?

 With proper resampling, this should be the case.

 Somebody clever could probably make this work.

DART Tutorial Index to Sections