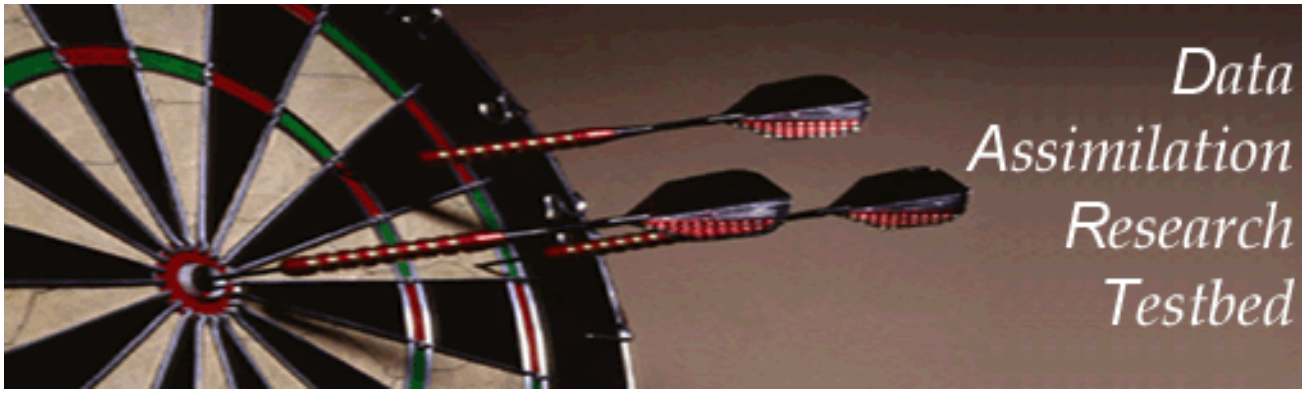
Data Assimilation Research Testbed Tutorial

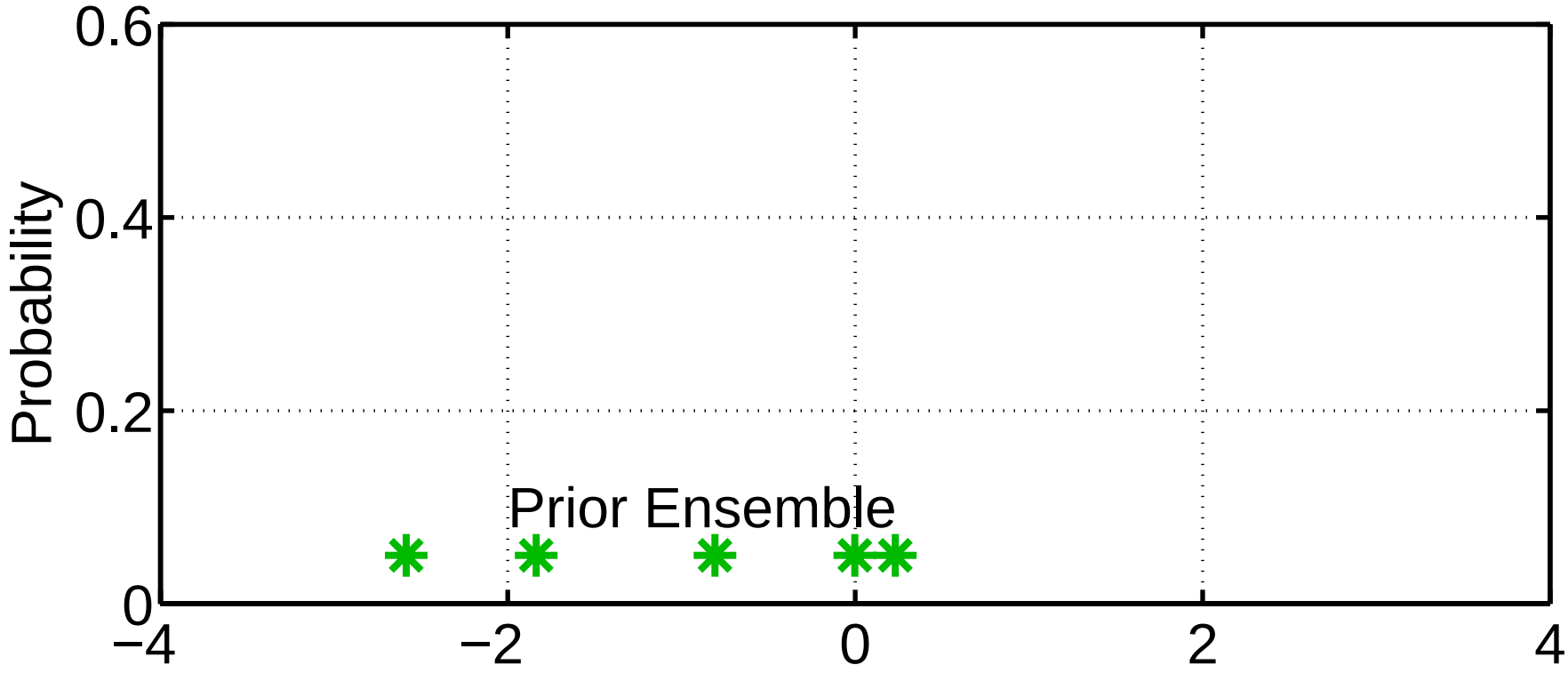


Section 6: Other Updates for An Observed Variable

Version 2.0: September, 2006

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

Ensemble filters: Prior is available as finite sample.

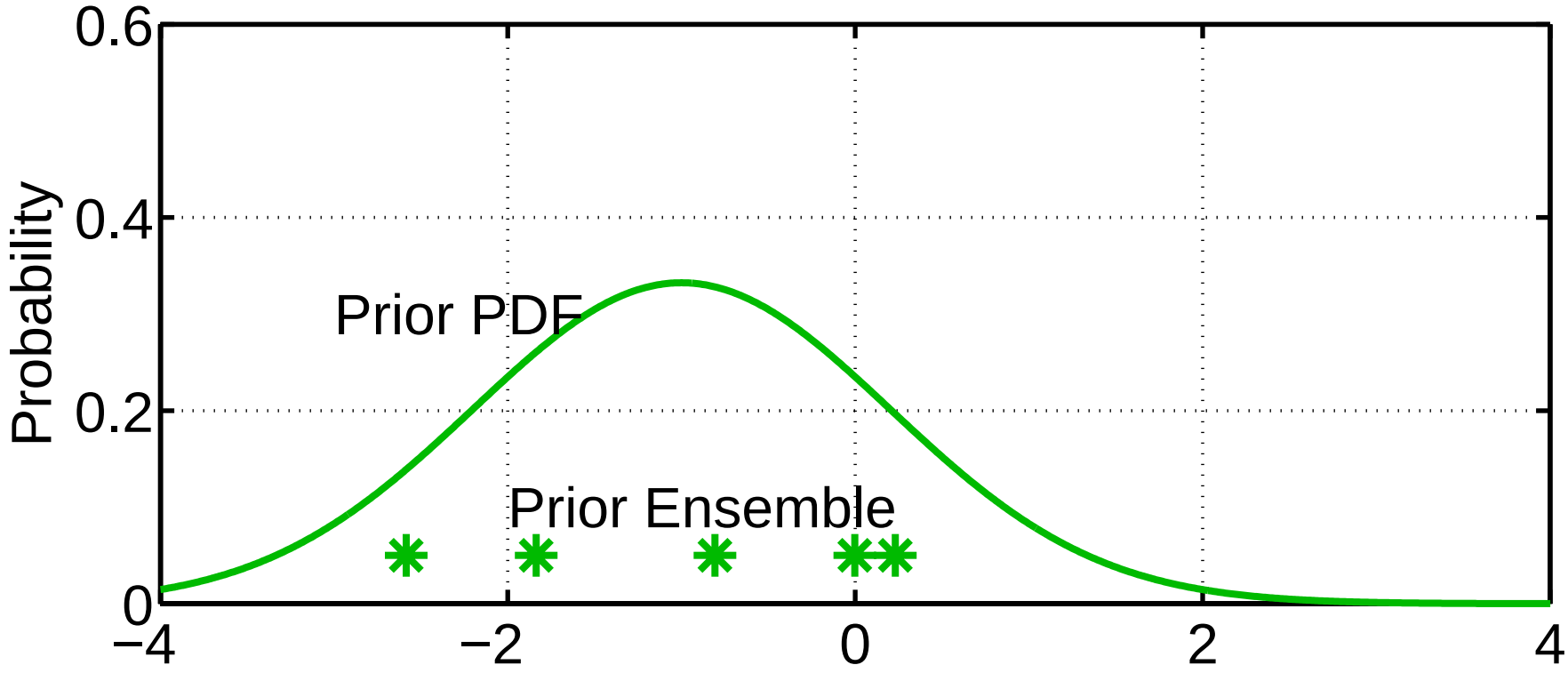


Don't know much about properties of this sample.

May naively assume it is random draw from 'truth'.

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

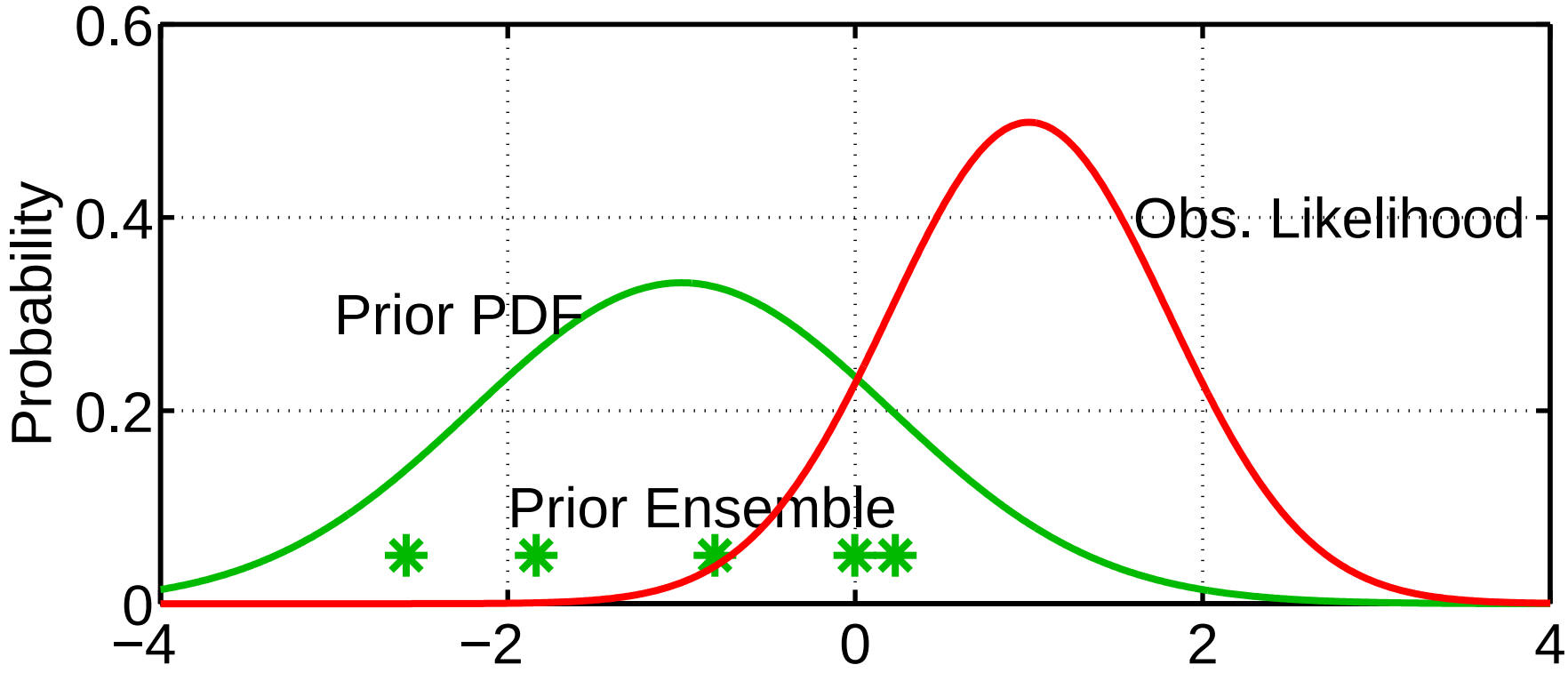
How can we take product of sample with continuous likelihood?



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

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

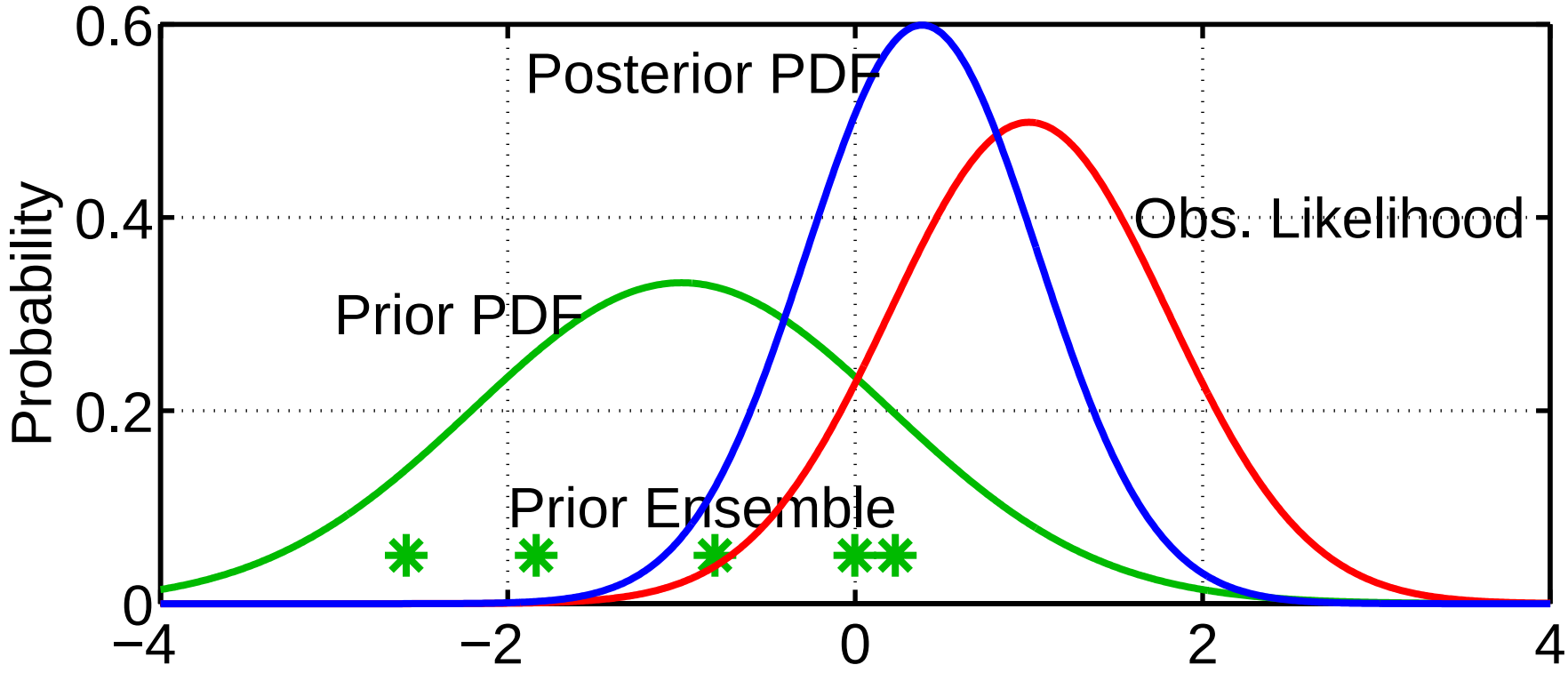
Observation likelihood usually continuous (nearly always Gaussian).



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

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

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

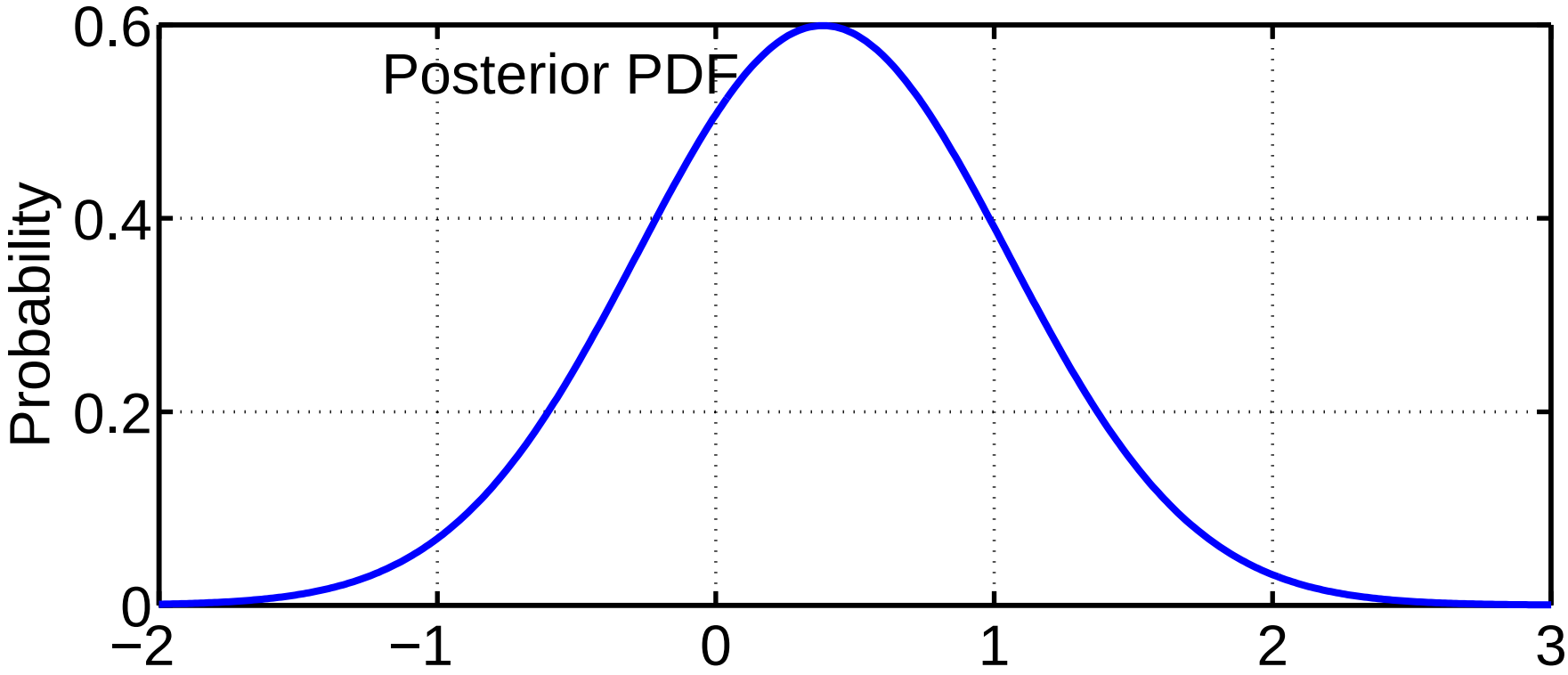


Computing continuous posterior is simple.

BUT, need to have a SAMPLE of this PDF.

Sampling Posterior PDF:

There are many ways to do this.



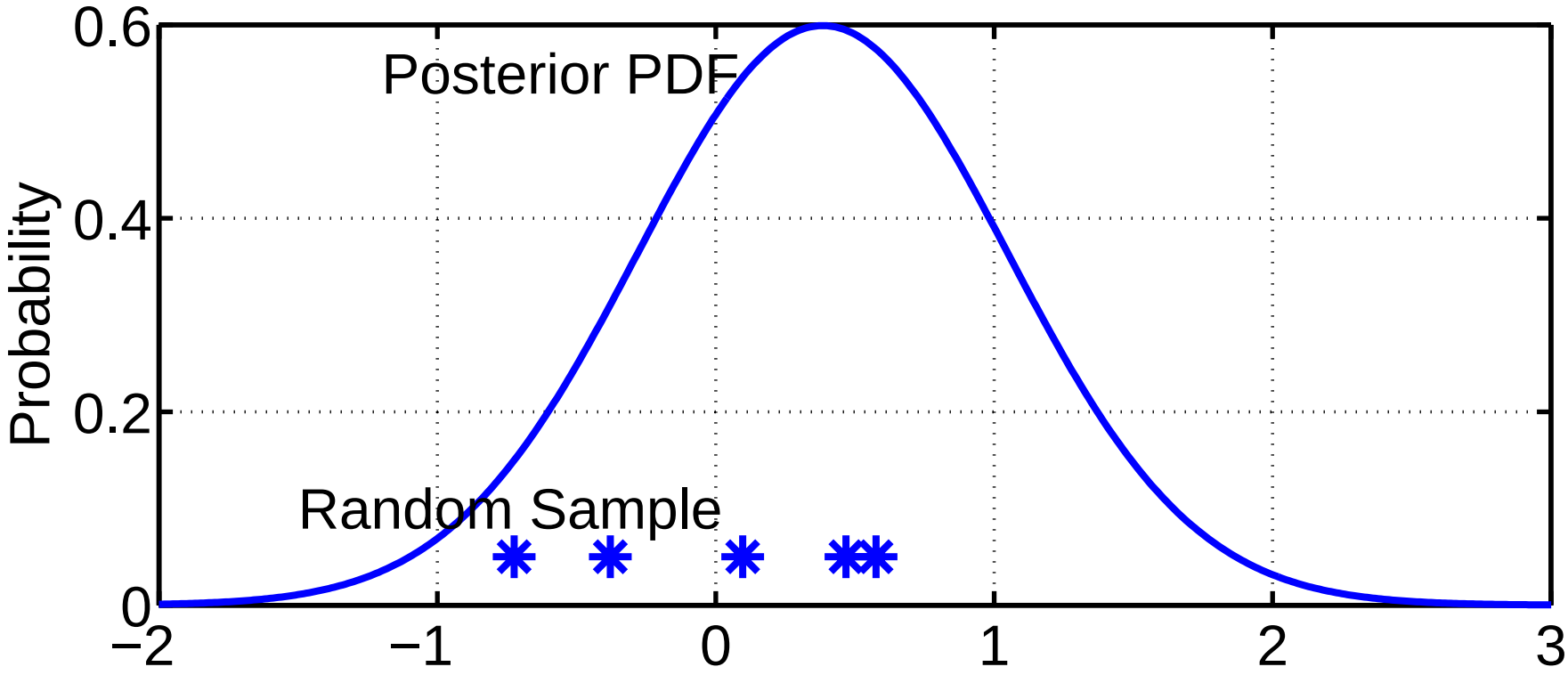
Exact properties of different methods may be unclear.

Trial and error still best way to see how they perform.

Will interact with properties of prediction models, etc.

Sampling Posterior PDF:

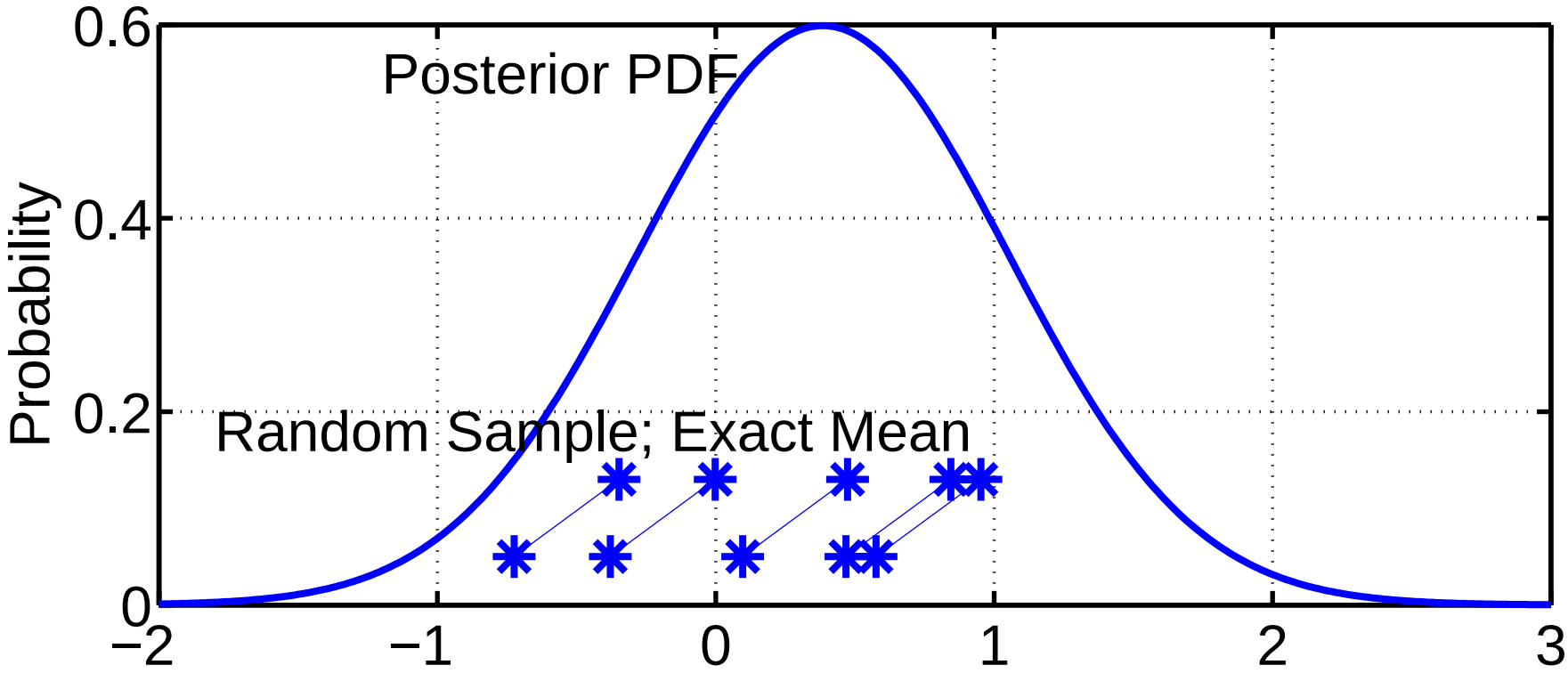
Just draw a random sample (filter_kind=5 in *assim_tools_nml*).



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).

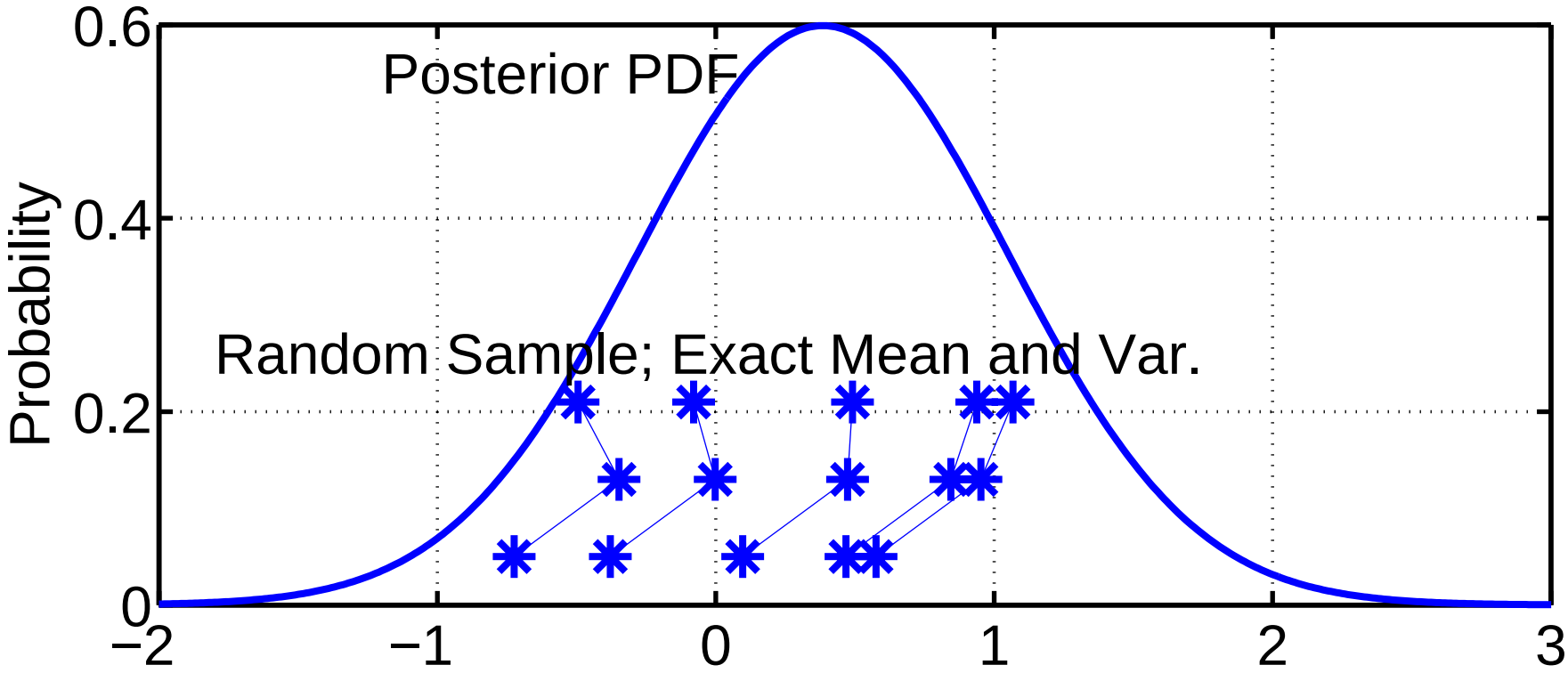


Can ‘play games’ with this sample to improve (modify) its properties.

Example: Adjust the mean of sample to be exact.

Sampling Posterior PDF:

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

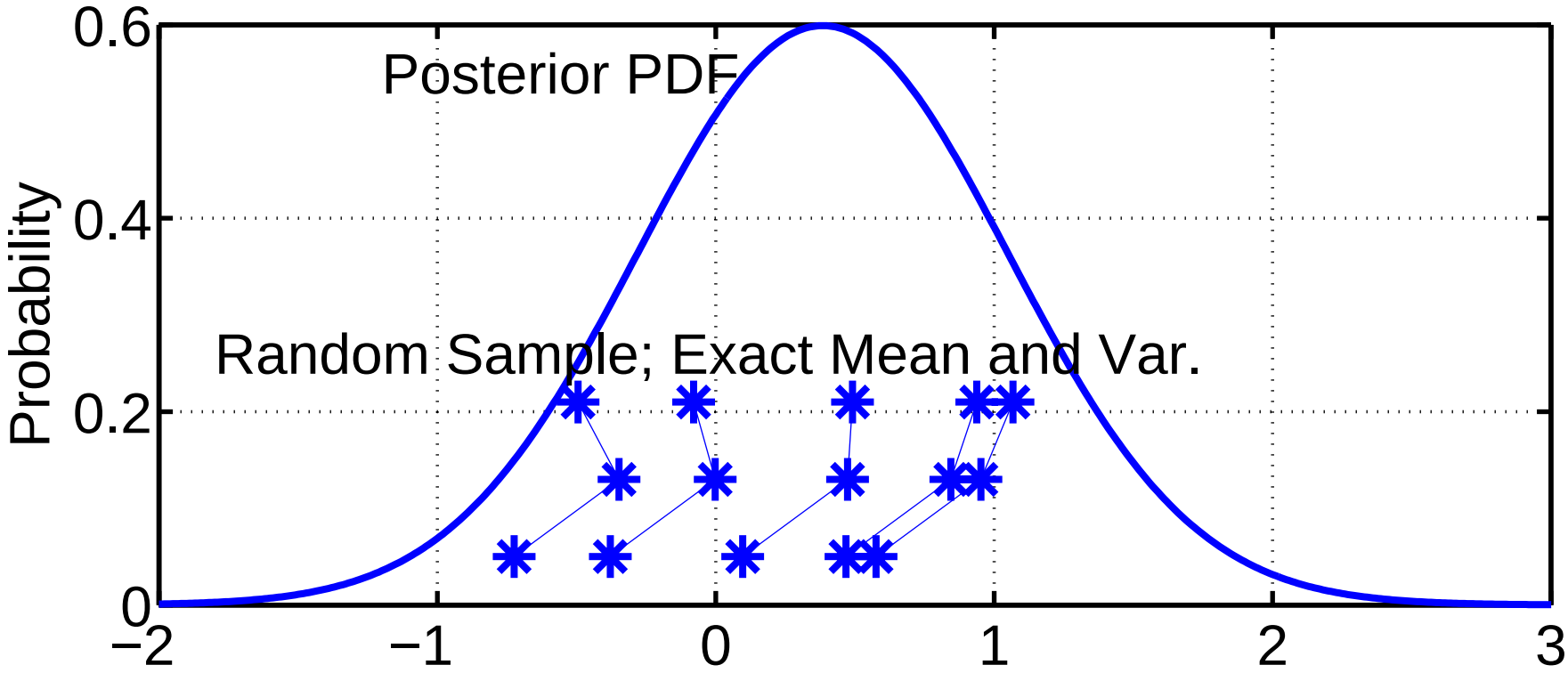


Can ‘play games’ with this sample to improve (modify) its properties.

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

Sampling Posterior PDF:

Just draw a random sample.

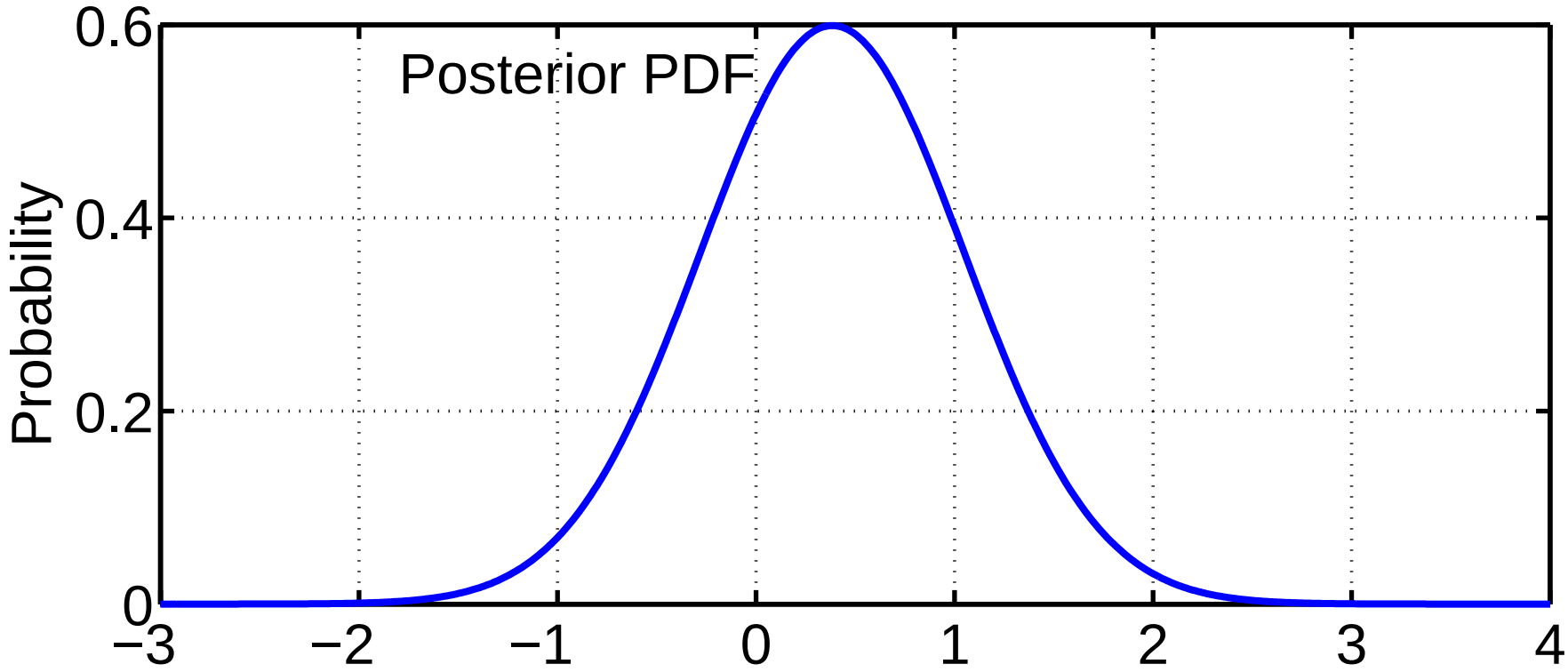


Might also want to eliminate rare extreme outliers.

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

Sampling Posterior PDF:

Construct a ‘deterministic’ sample with certain features.

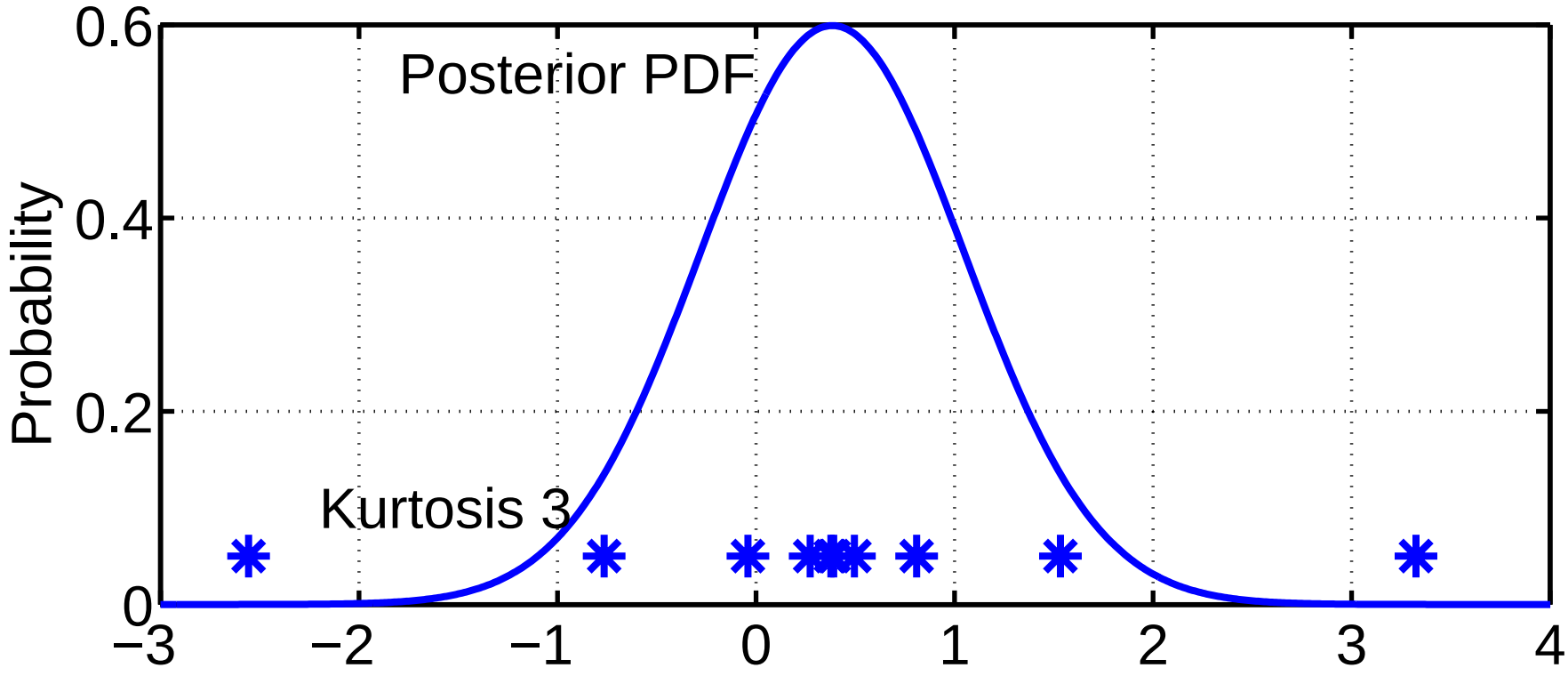


For instance: Sample could have exact mean and variance.

This is insufficient to constrain ensemble, need other constraints.

Sampling Posterior PDF:

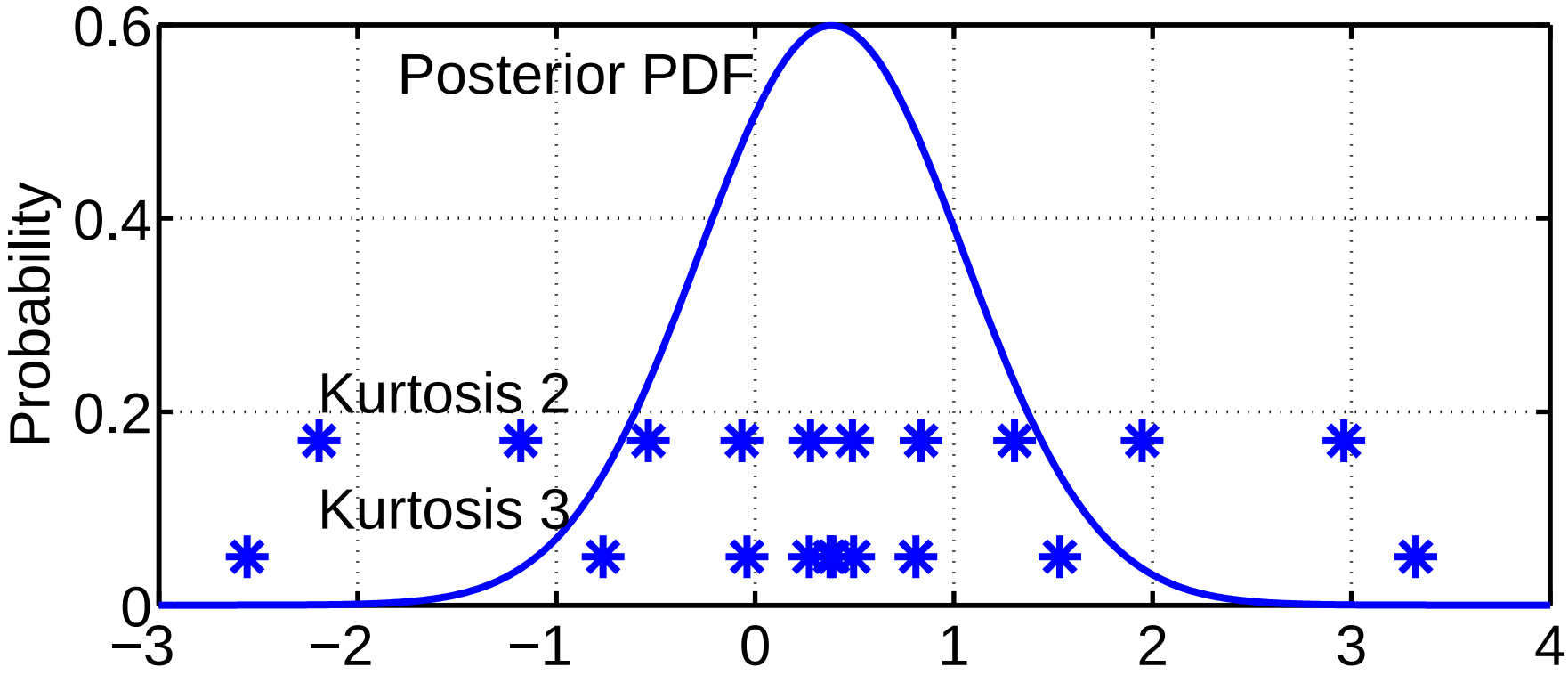
Construct a ‘deterministic’ sample with certain features
(*filter_kind=6* in *assim_tools_nml*; manually adjust kurtosis).



Example: Exact sample mean and variance.
Sample kurtosis is 3 (expected value for Gaussian in large sample limit)
(Constructed by starting uniformly spaced and adjusting quadratically).

Sampling Posterior PDF:

Construct a ‘deterministic’ sample with certain feature
(*filter_kind=6* in *assim_tools_nml*; manually adjust kurtosis).



Example: Exact sample mean and variance.

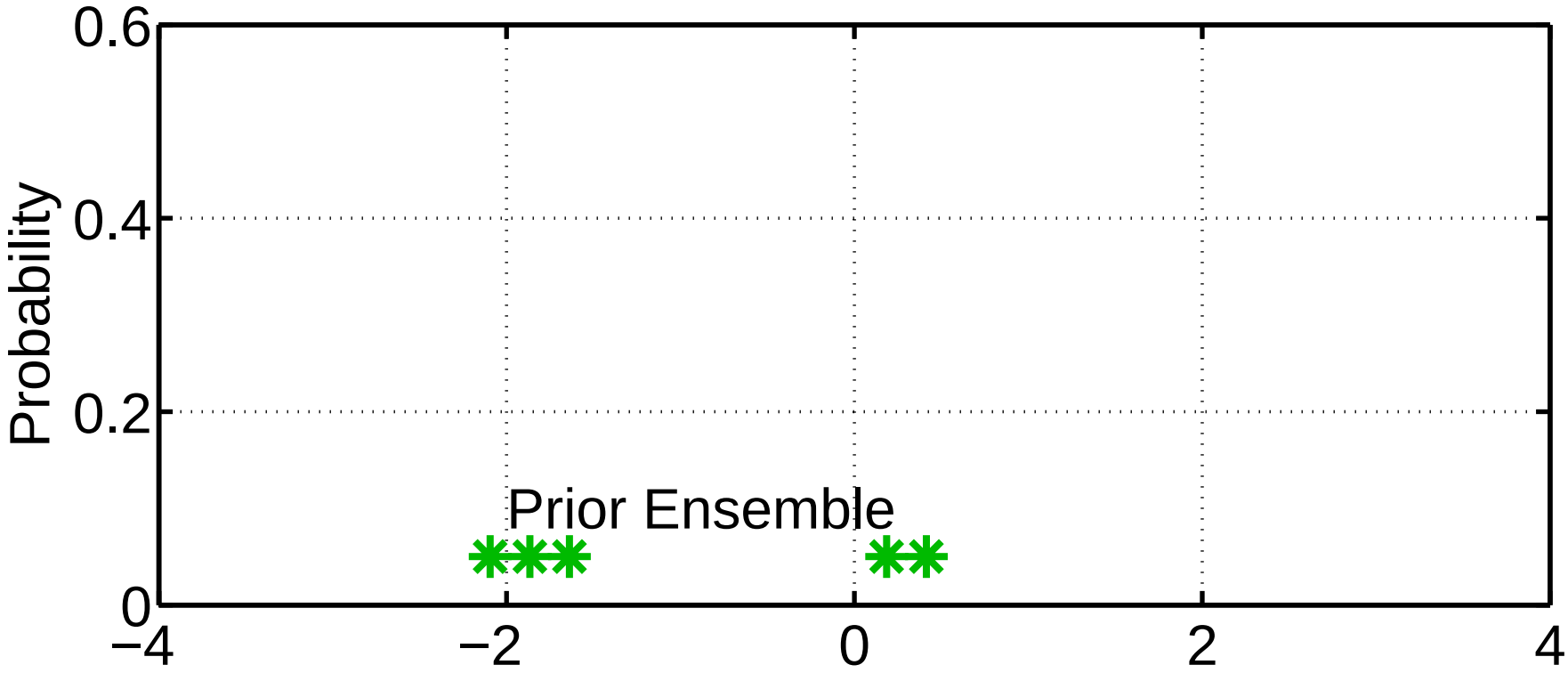
Sample kurtosis 2: less extreme outliers, less dense near mean.

Avoiding outliers might be nice in certain applications.

Sampling heavily near mean might be nice.

Sampling Posterior PDF:

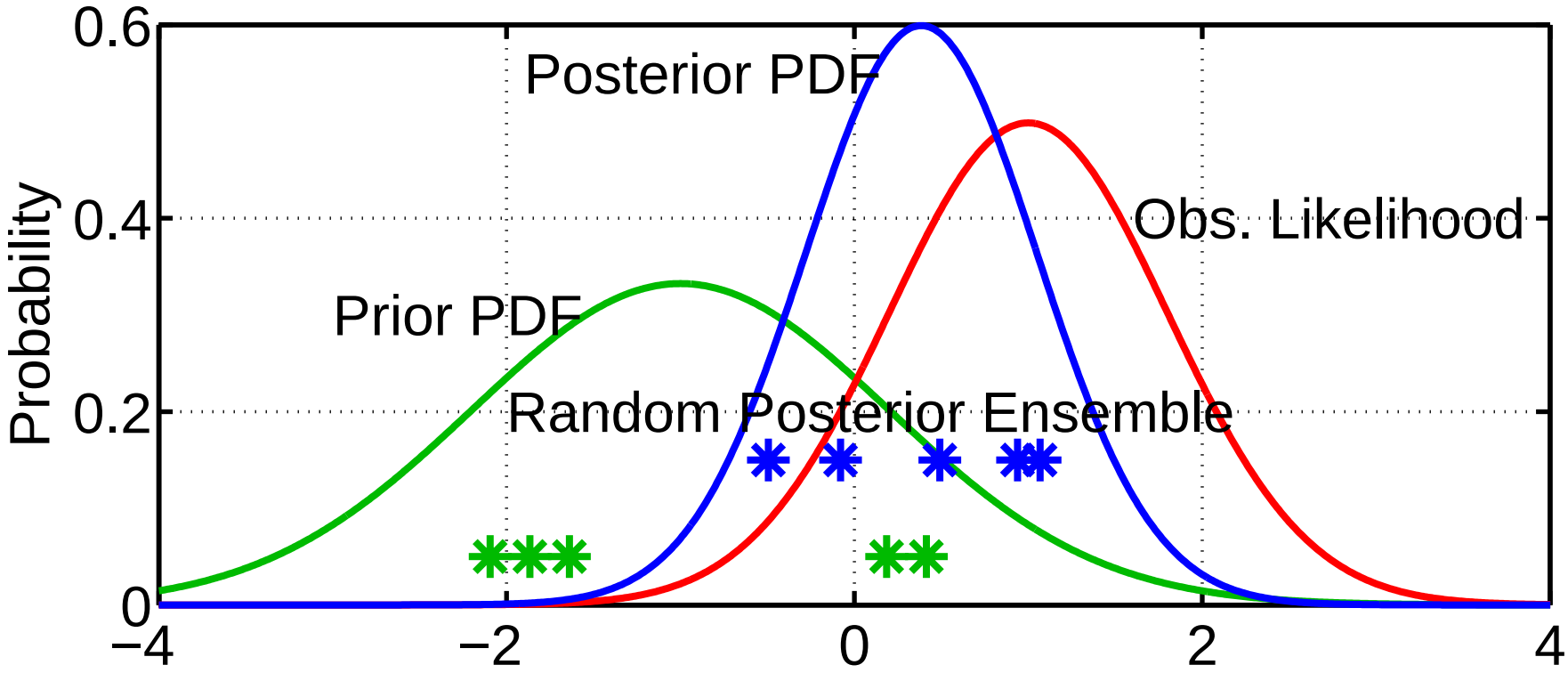
First two methods depend only on mean and variance of prior sample.



Example: Suppose prior sample is (significantly) bimodal?

Sampling Posterior PDF:

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



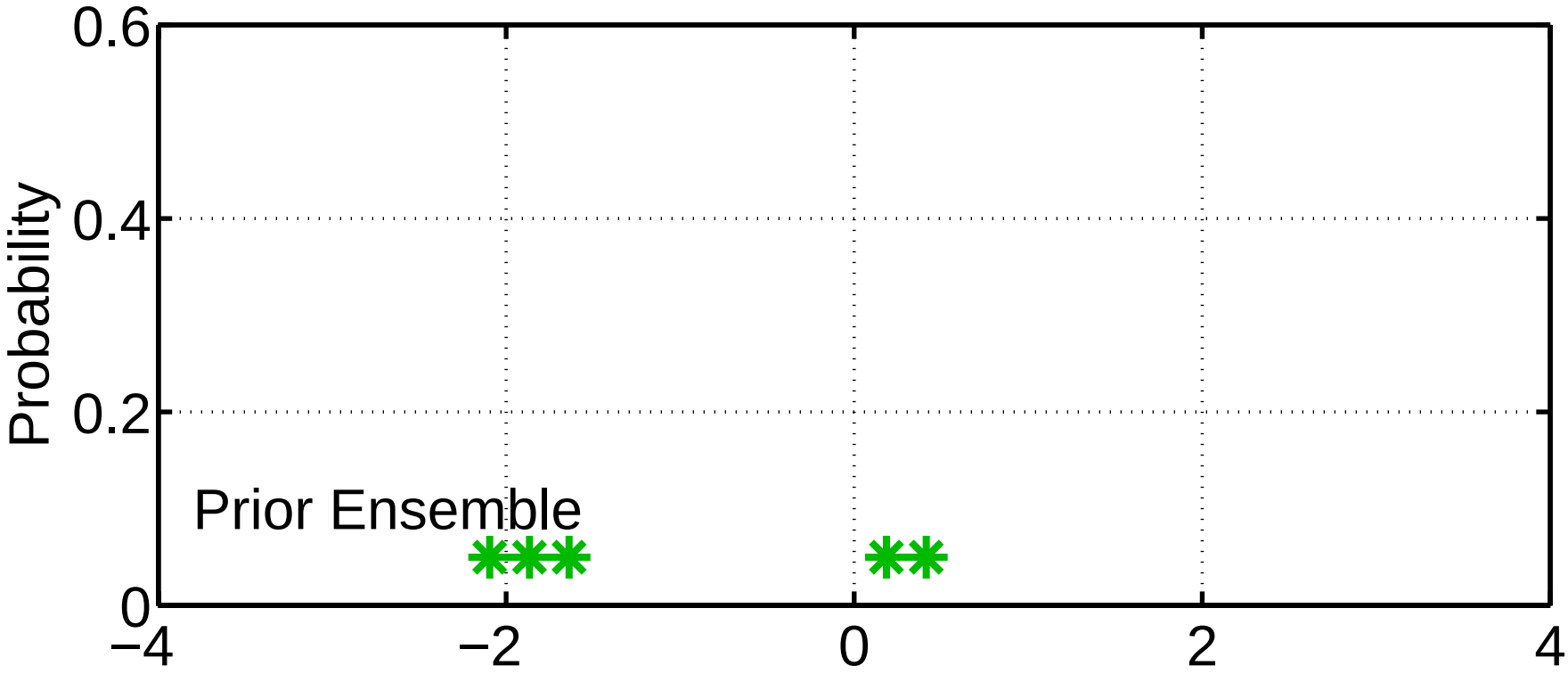
Example: Suppose prior sample is (significantly) bimodal?

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*).

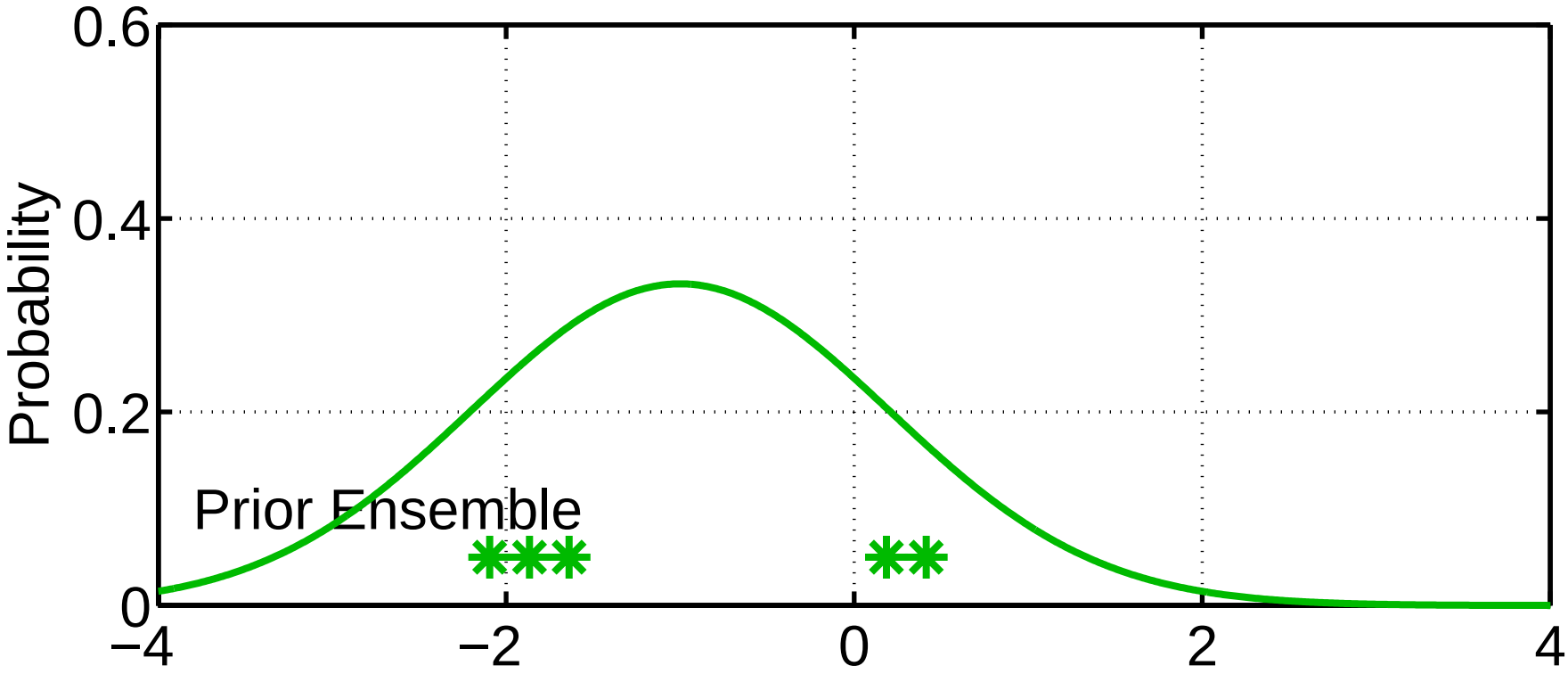


'Classical' Monte Carlo Algorithm for Data Assimilation.

Warning: earliest refs have incorrect algorithm (more in a minute).

Ensemble Filter Algorithms:

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

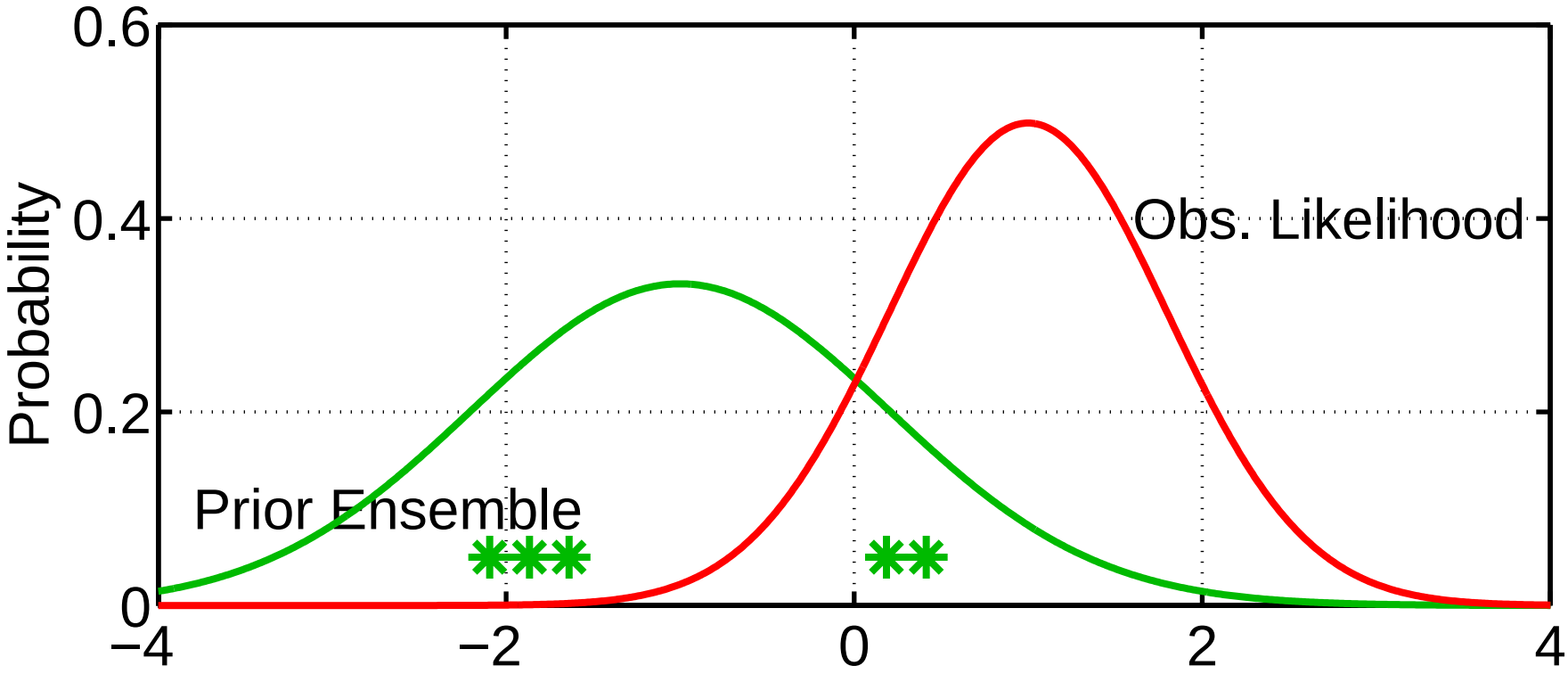


Again, fit a Gaussian to sample.

Are there ways to do this without computing prior sample stats?

Ensemble Filter Algorithms:

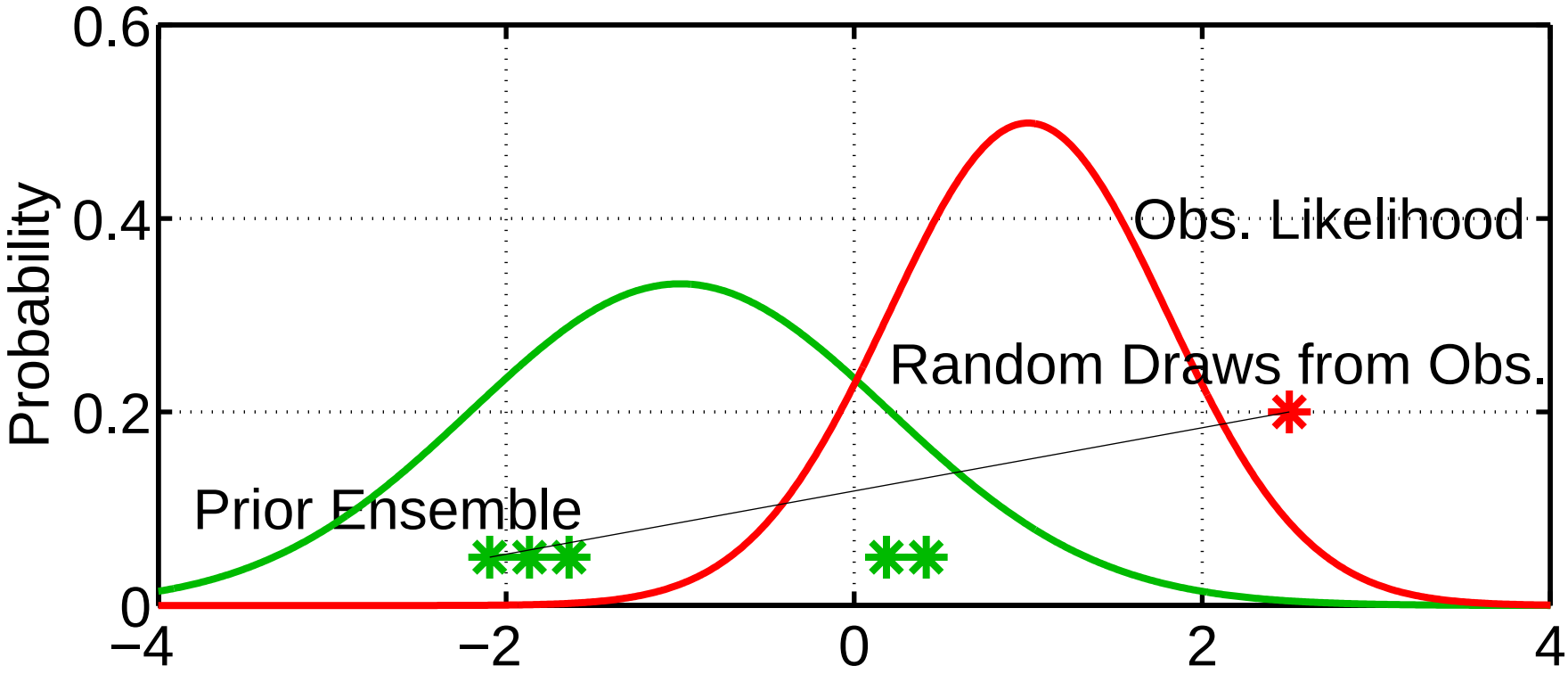
Ensemble Kalman Filter (EnKF) (*filter_kind=2* in *assim_tools_nml*).



Again, fit a Gaussian to sample.

Ensemble Filter Algorithms:

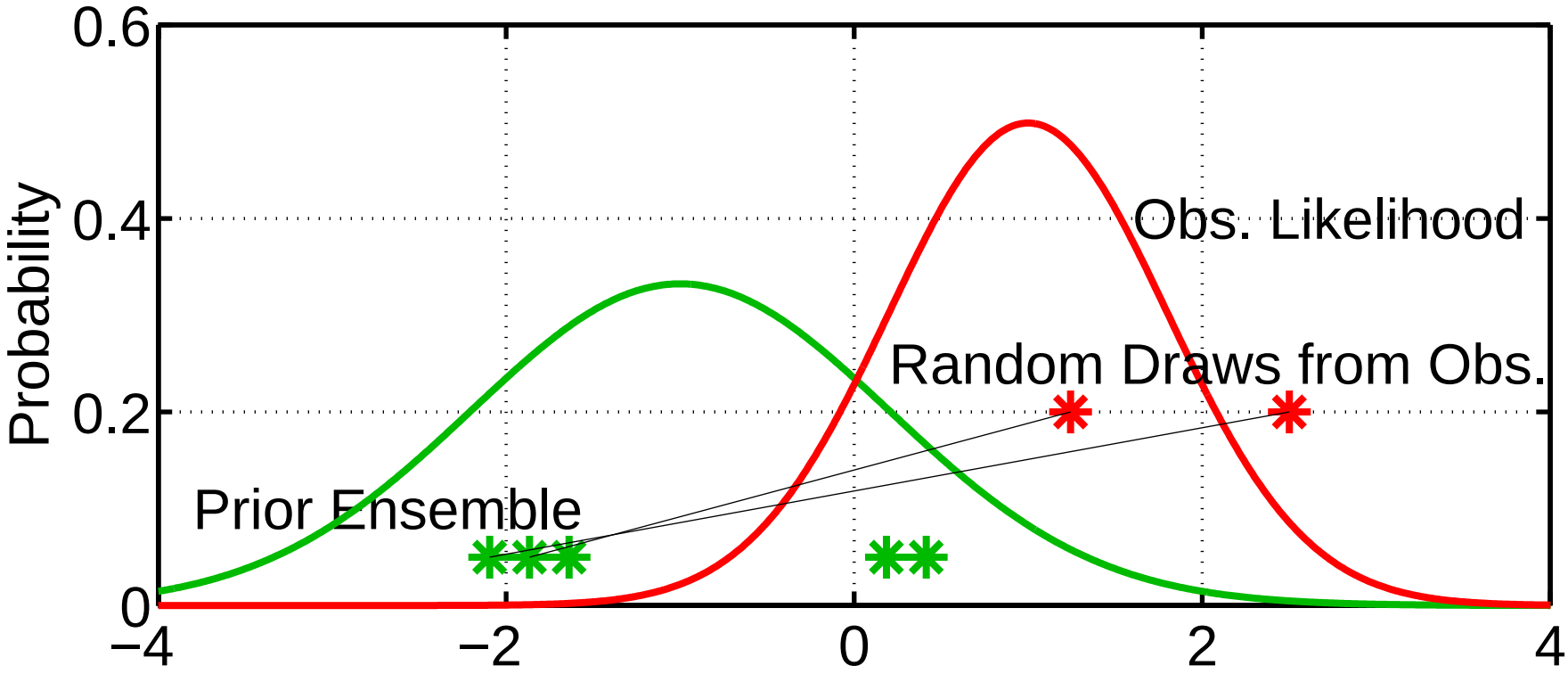
Ensemble Kalman Filter (EnKF) (*filter_kind=2* in *assim_tools_nml*).



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

Ensemble Filter Algorithms:

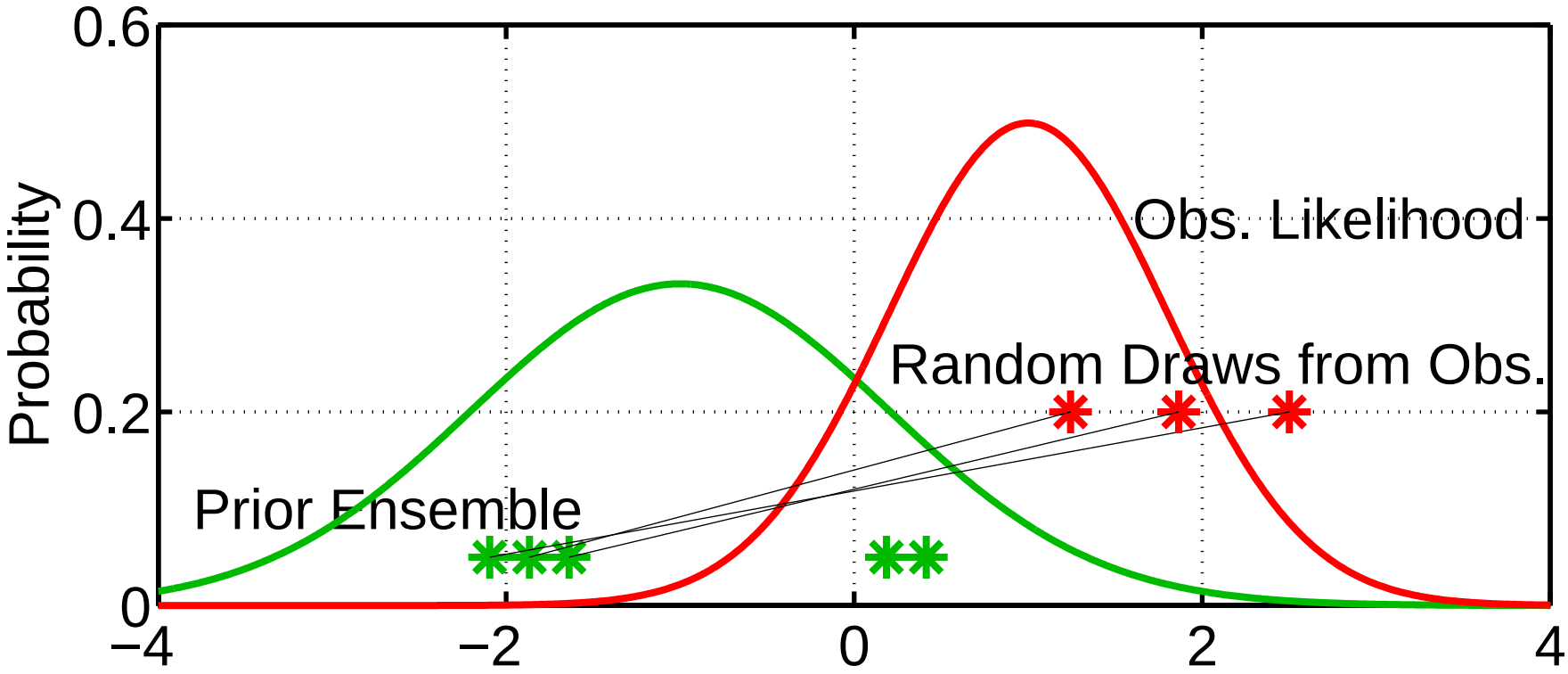
Ensemble Kalman Filter (EnKF) (*filter_kind=2* in *assim_tools_nml*).



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:

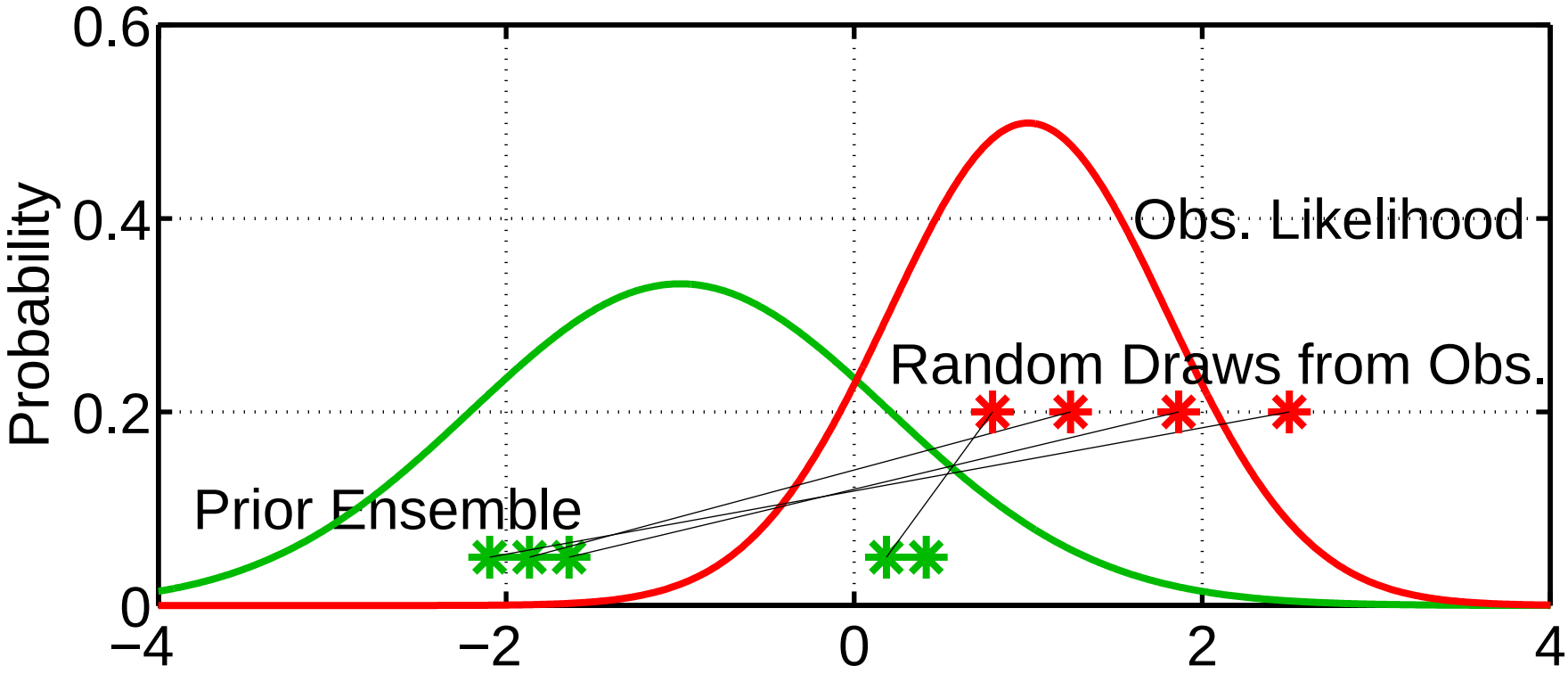
Ensemble Kalman Filter (EnKF) (*filter_kind=2* in *assim_tools_nml*).



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*).



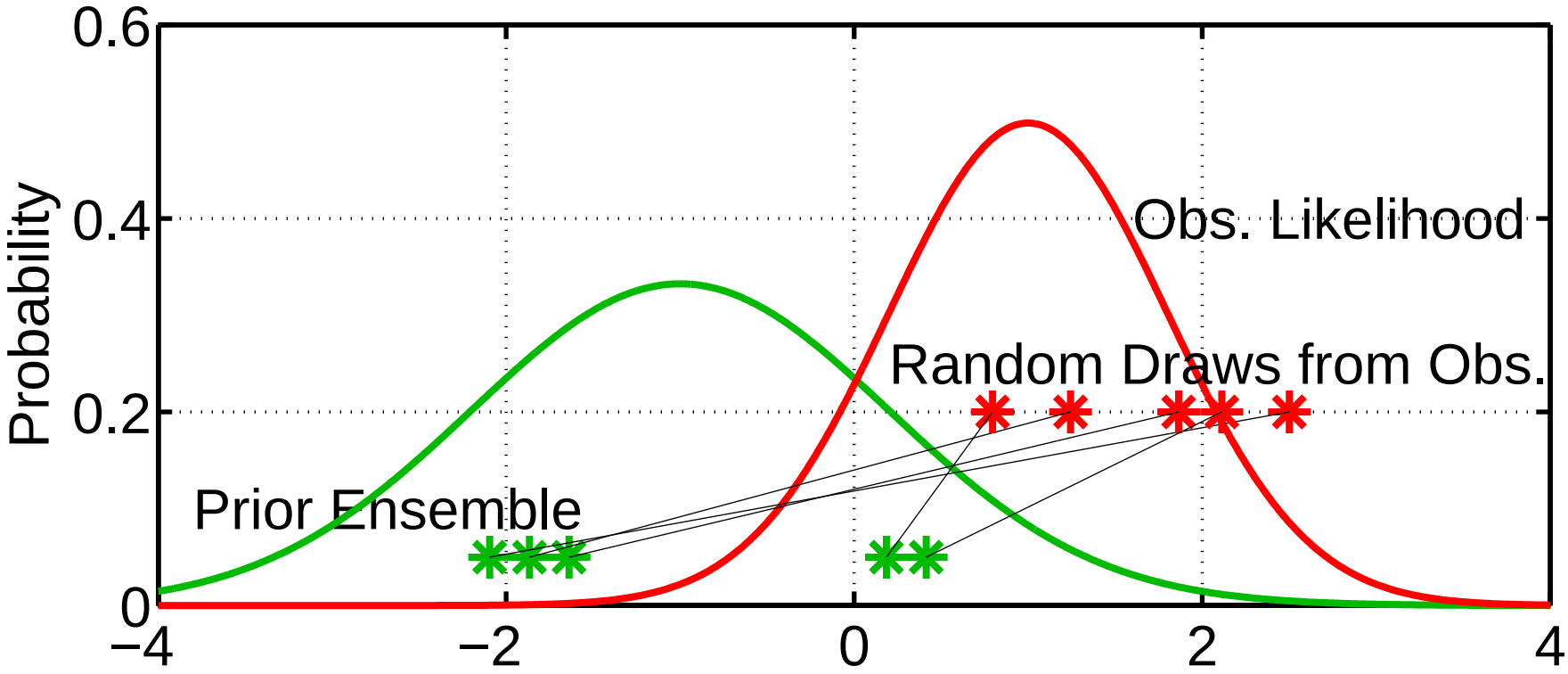
Proceed to associate a random draw from obs. with each prior sample.

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

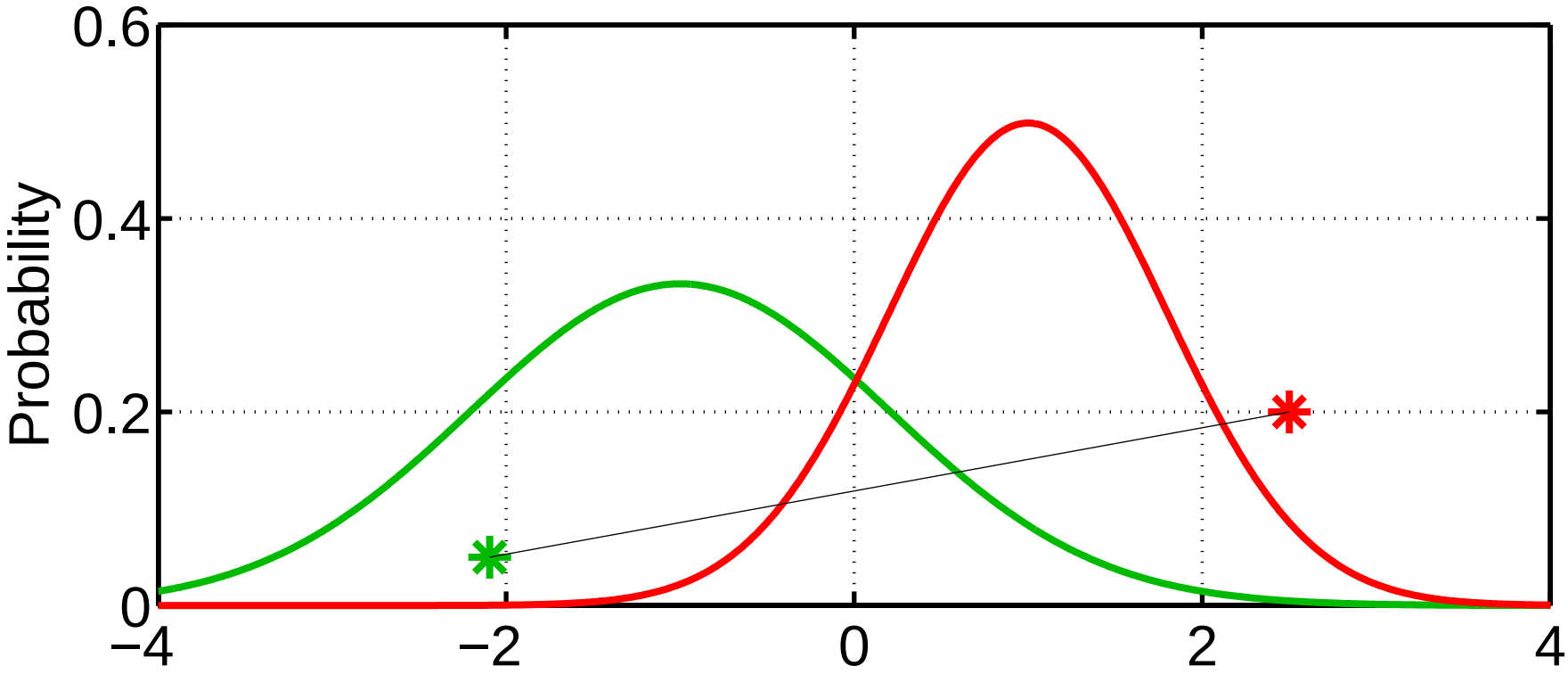
Adjusting the mean of obs. sample to be exact improves performance.

Adjusting the variance may further improve performance.

Outliers are potential problem, but can be removed.

Ensemble Filter Algorithms:

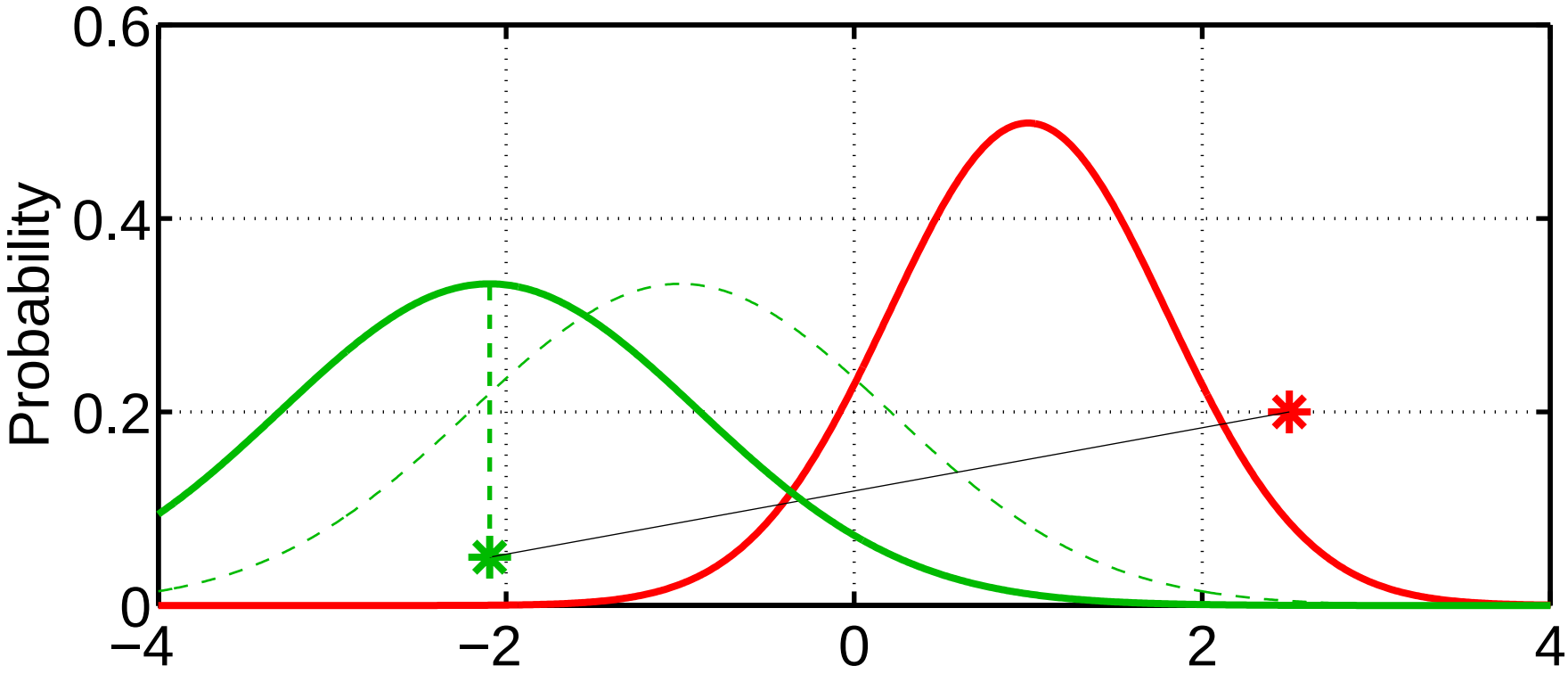
Ensemble Kalman Filter (EnKF) (*filter_kind=2* in *assim_tools_nml*).



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

Ensemble Filter Algorithms:

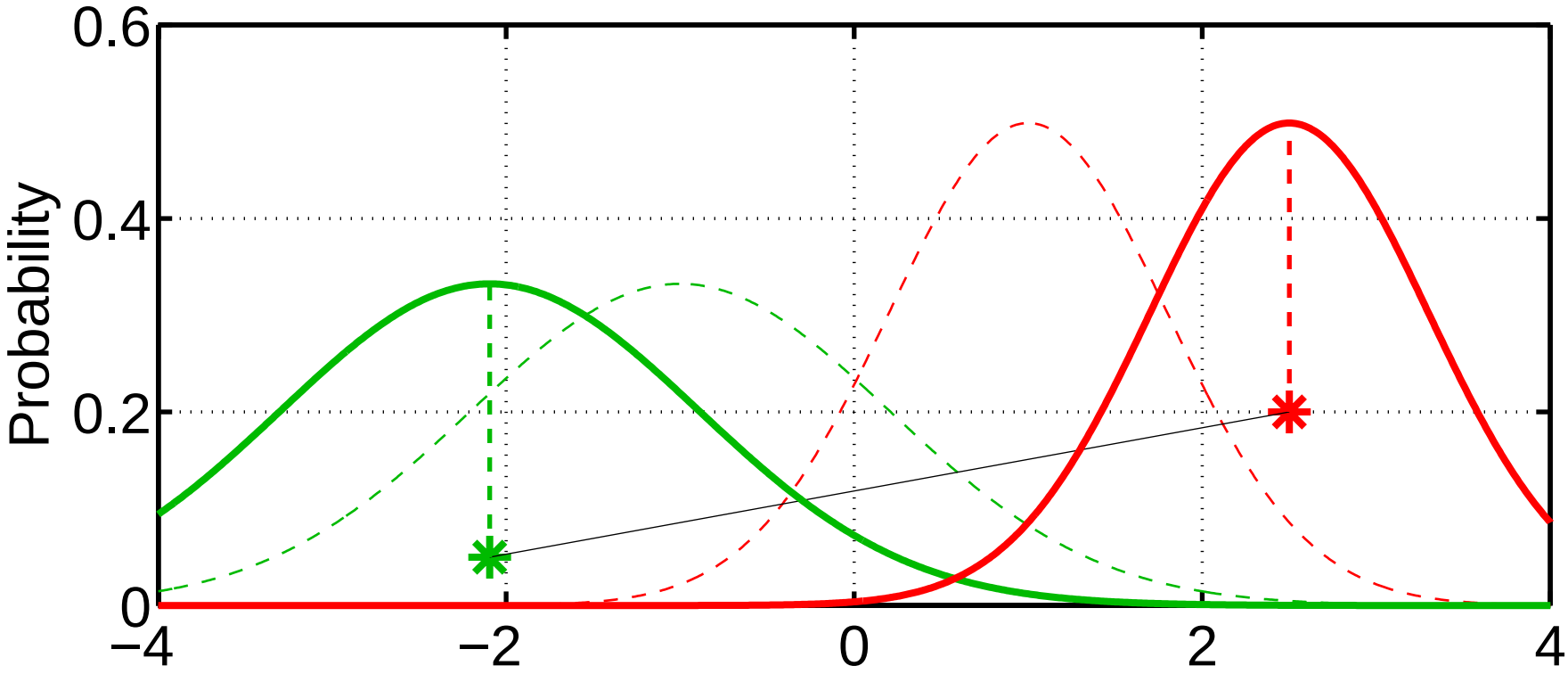
Ensemble Kalman Filter (EnKF) (*filter_kind=2* in *assim_tools_nml*).



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*).

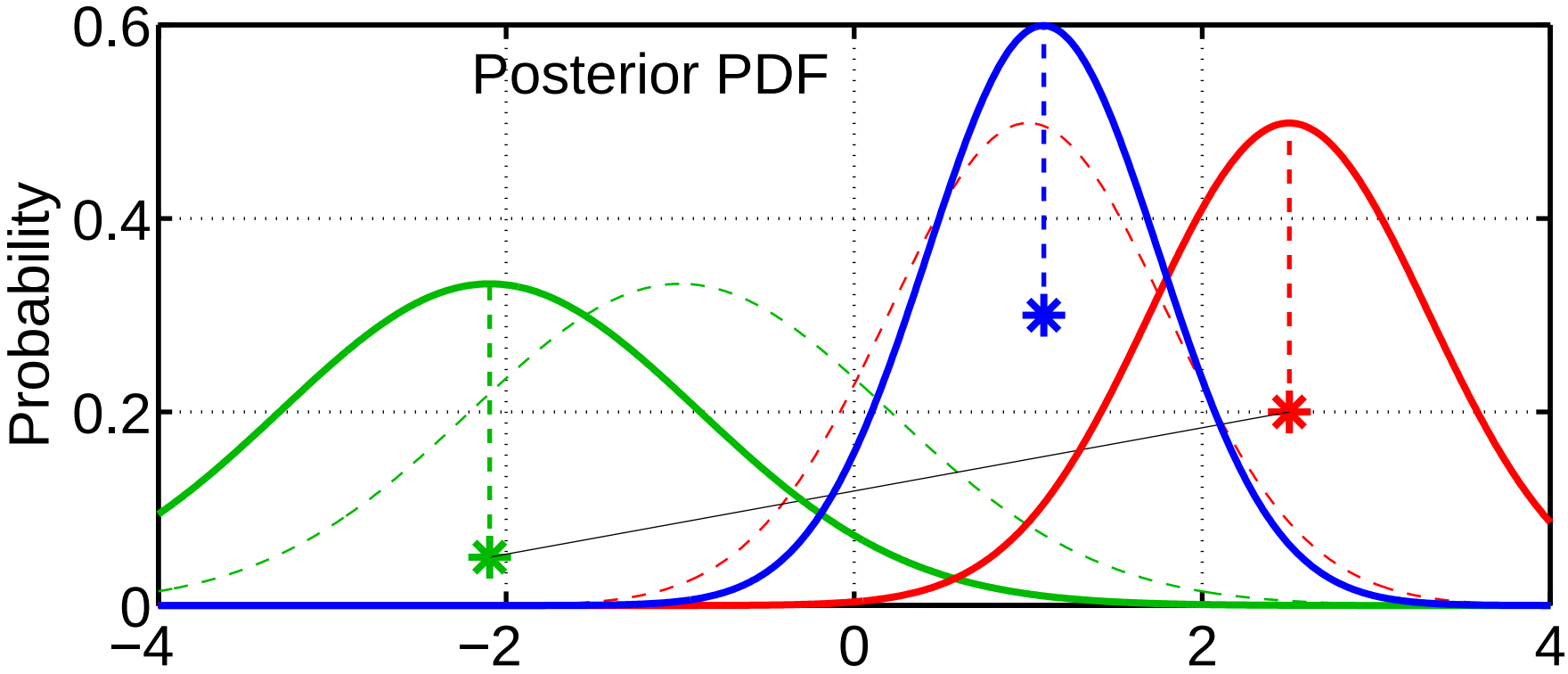


Prior sample standard deviation still measures uncertainty of prior mean estimate.

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

Ensemble Filter Algorithms:

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

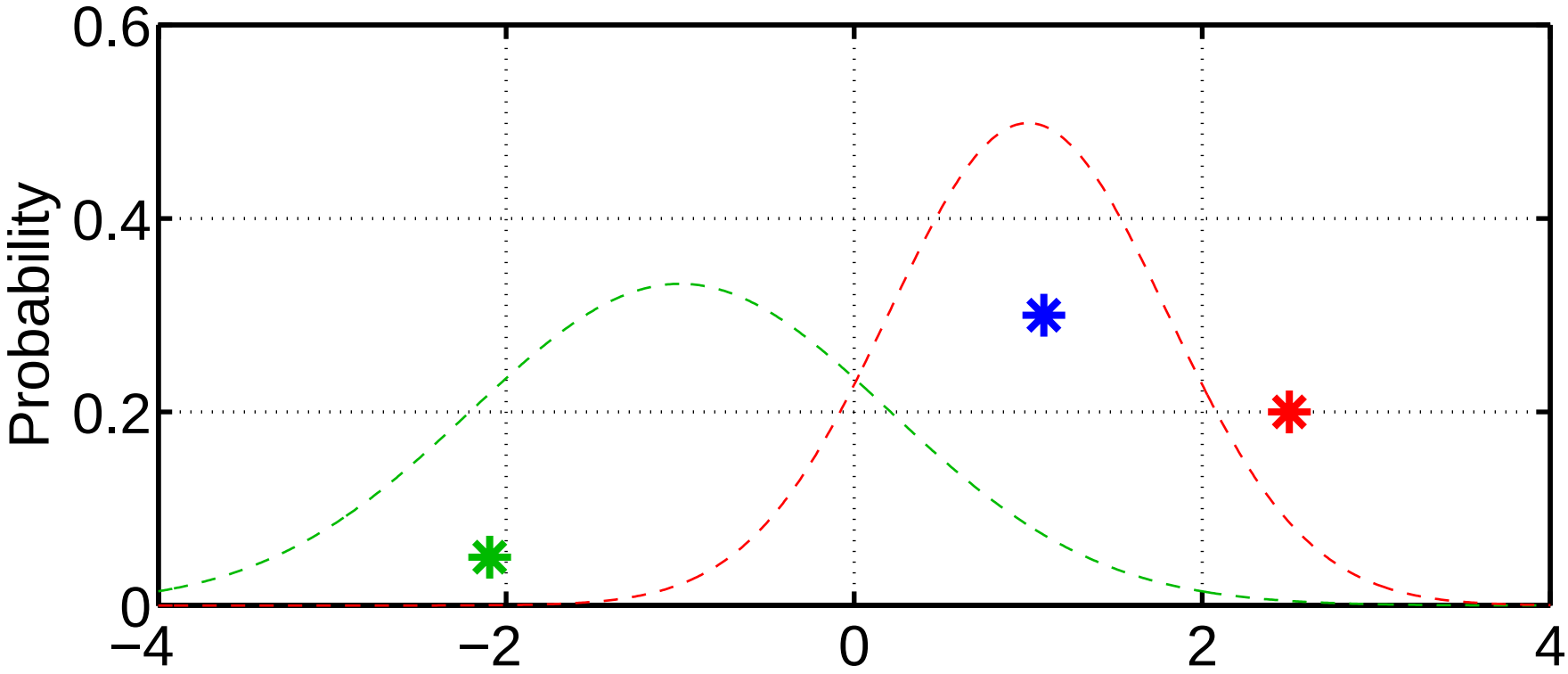


Take product of the prior/obs distributions for first sample.

This is standard Gaussian product.

Ensemble Filter Algorithms:

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

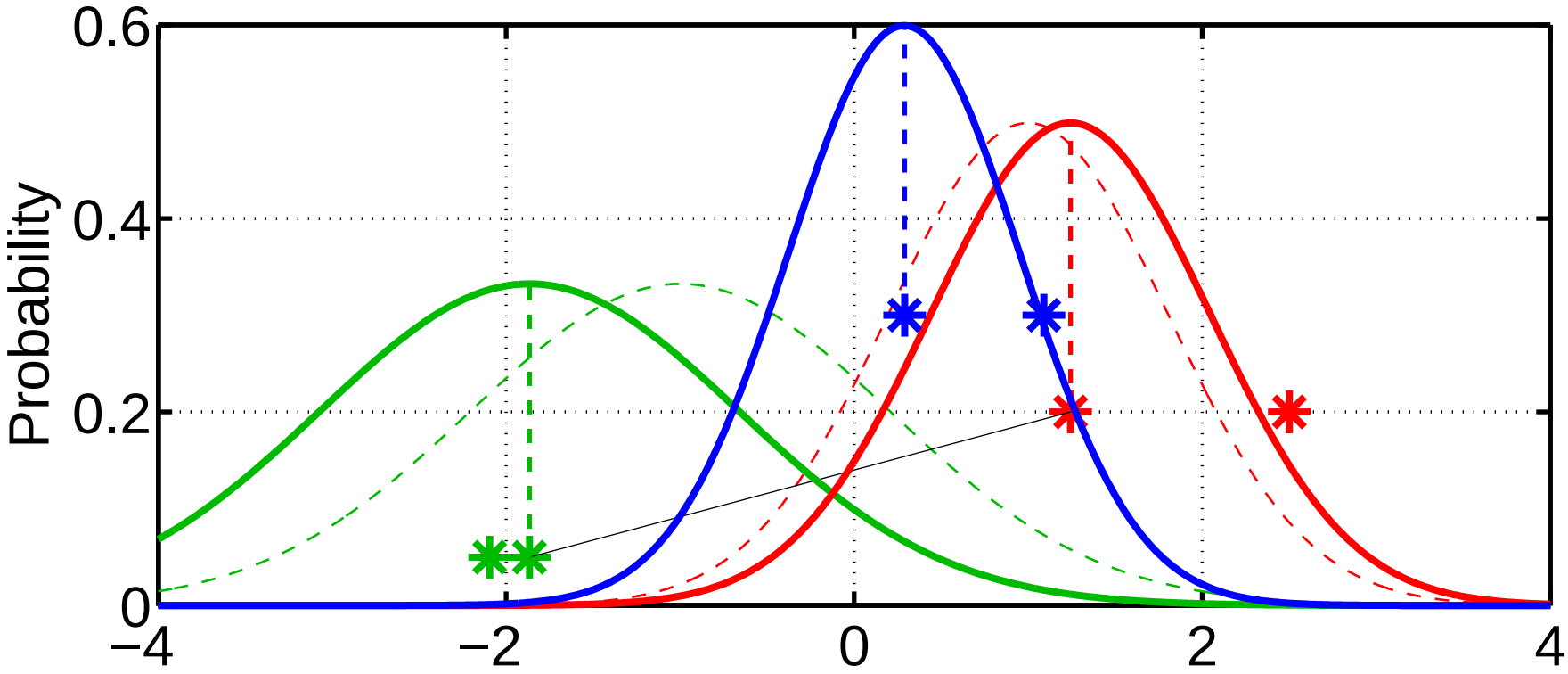


Mean of product is random sample of posterior.

Product of random samples is random sample of product.

Ensemble Filter Algorithms:

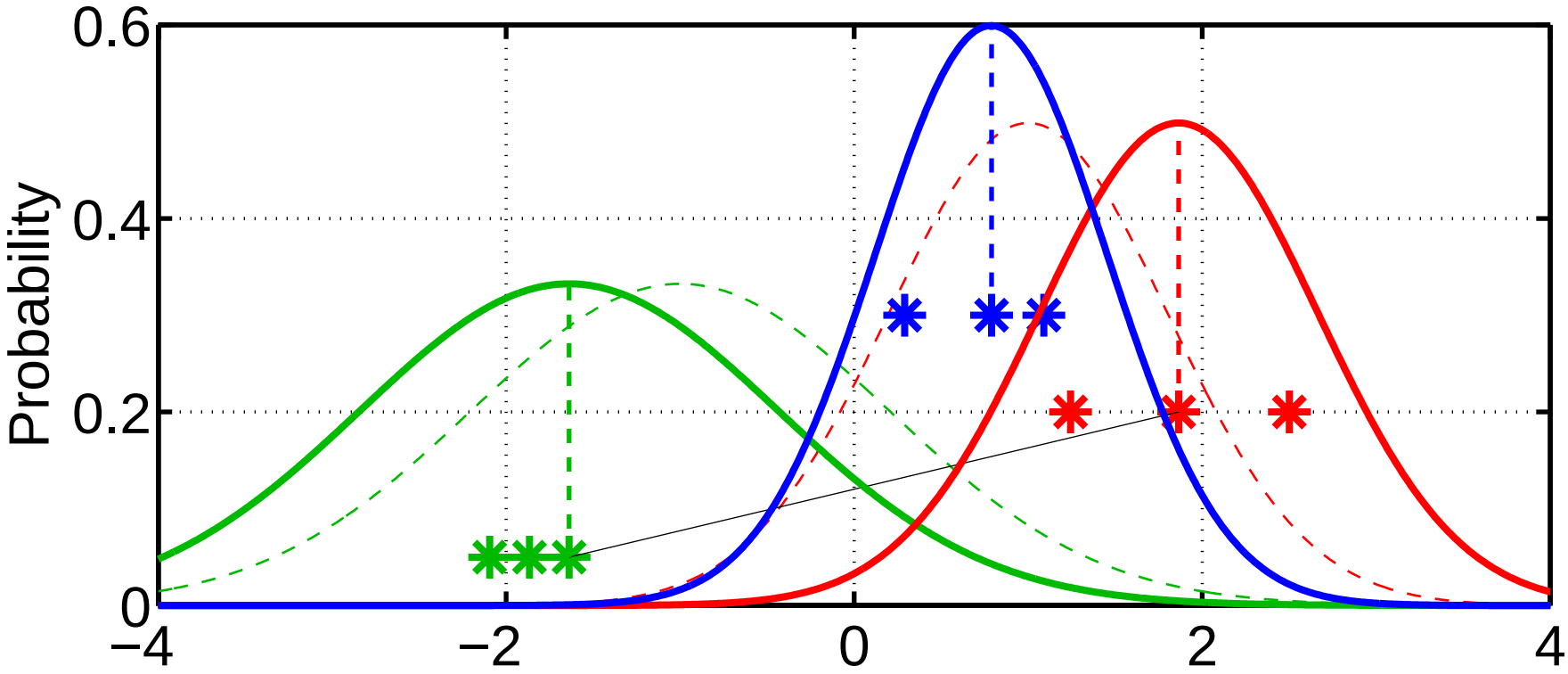
Ensemble Kalman Filter (EnKF) (*filter_kind=2* in *assim_tools_nml*).



Repeat this operation for each joint prior pair.

Ensemble Filter Algorithms:

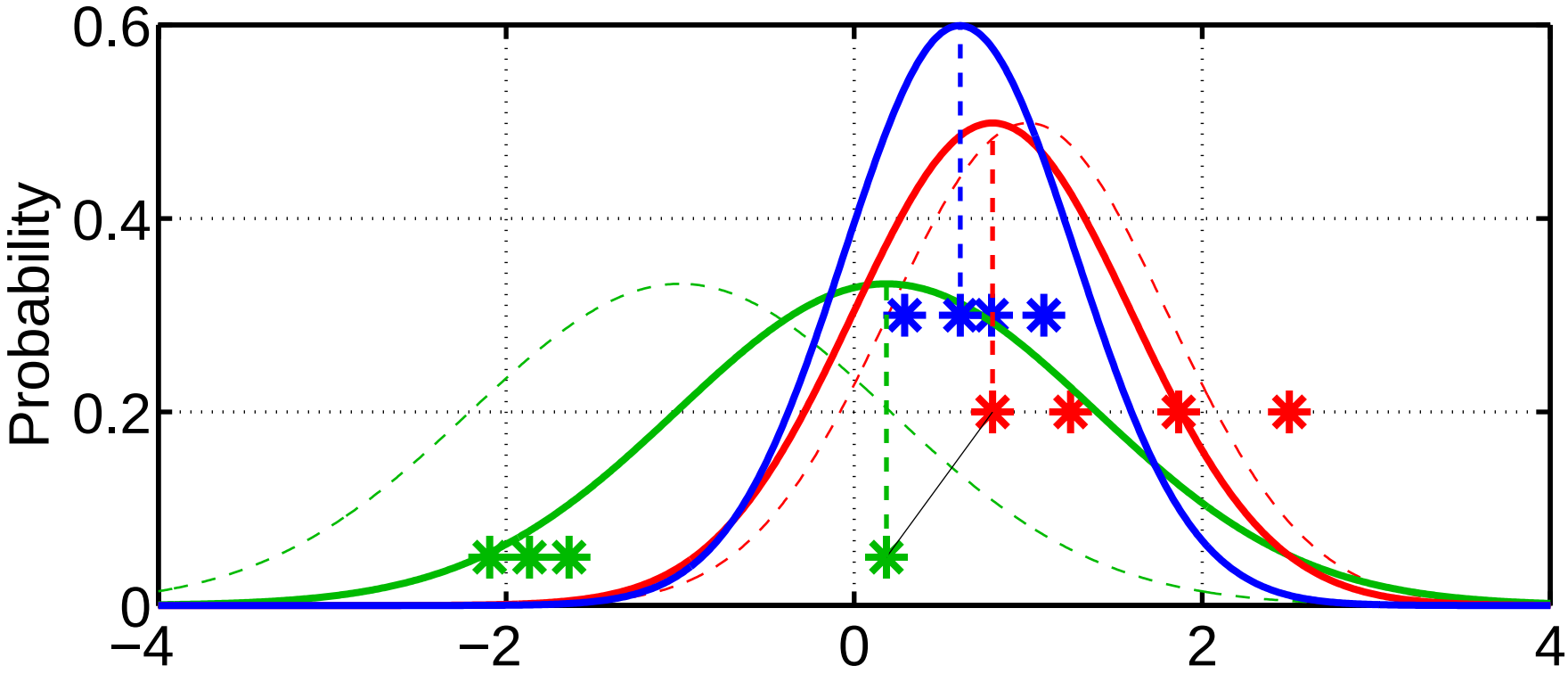
Ensemble Kalman Filter (EnKF) (*filter_kind=2* in *assim_tools_nml*).



Repeat this operation for each joint prior pair.

Ensemble Filter Algorithms:

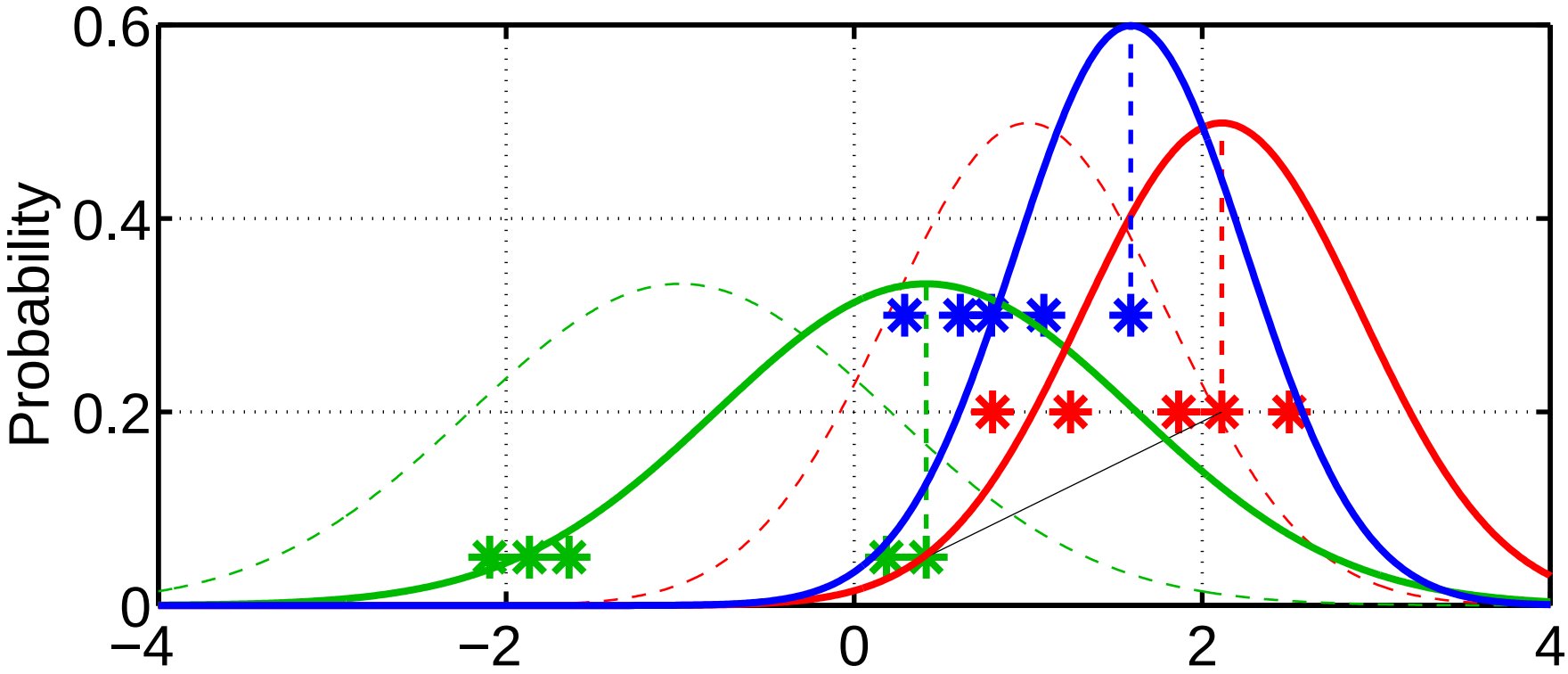
Ensemble Kalman Filter (EnKF) (*filter_kind=2* in *assim_tools_nml*).



Repeat this operation for each joint prior pair.

Ensemble Filter Algorithms:

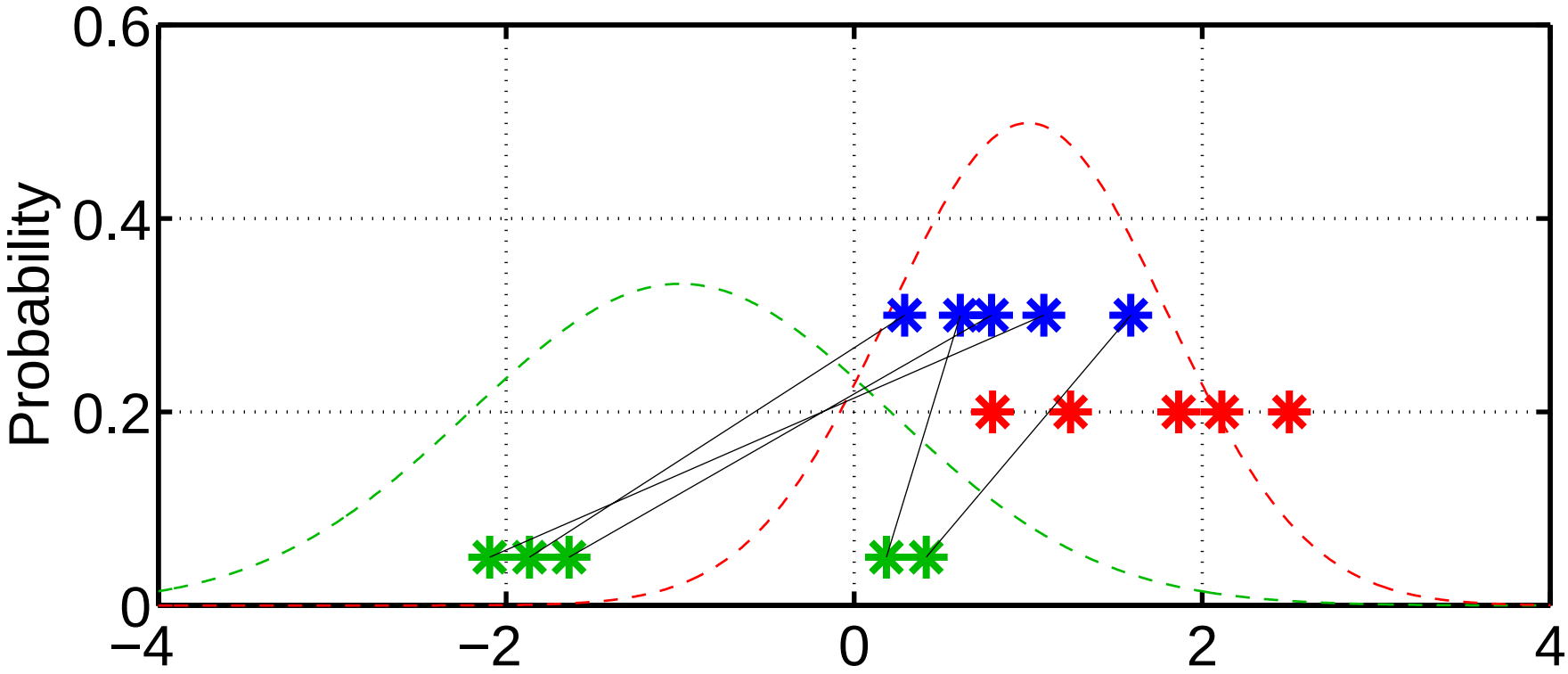
Ensemble Kalman Filter (EnKF) (*filter_kind=2* in *assim_tools_nml*).



Repeat this operation for each joint prior pair.

Ensemble Filter Algorithms:

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



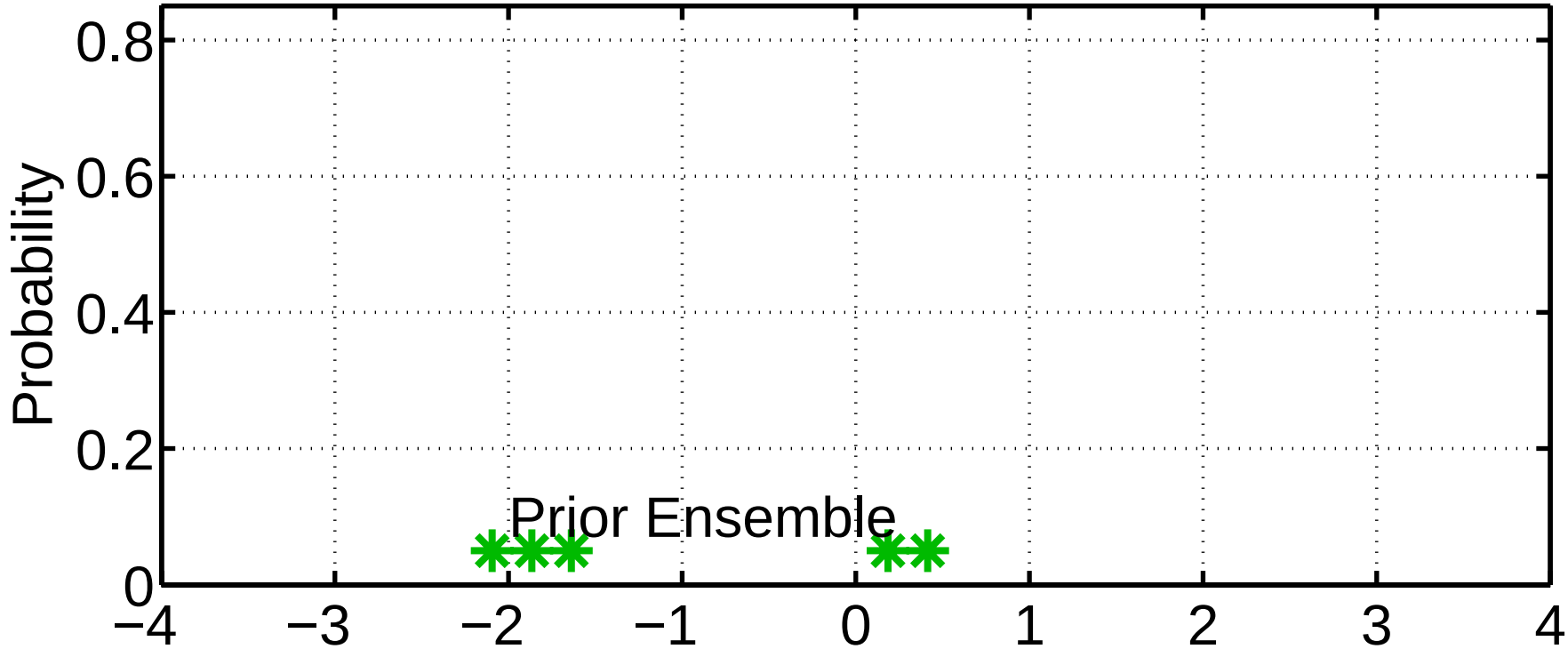
Posterior sample maintains much of prior sample structure.

(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 (filter_kind=3 in assim_tools_nml).

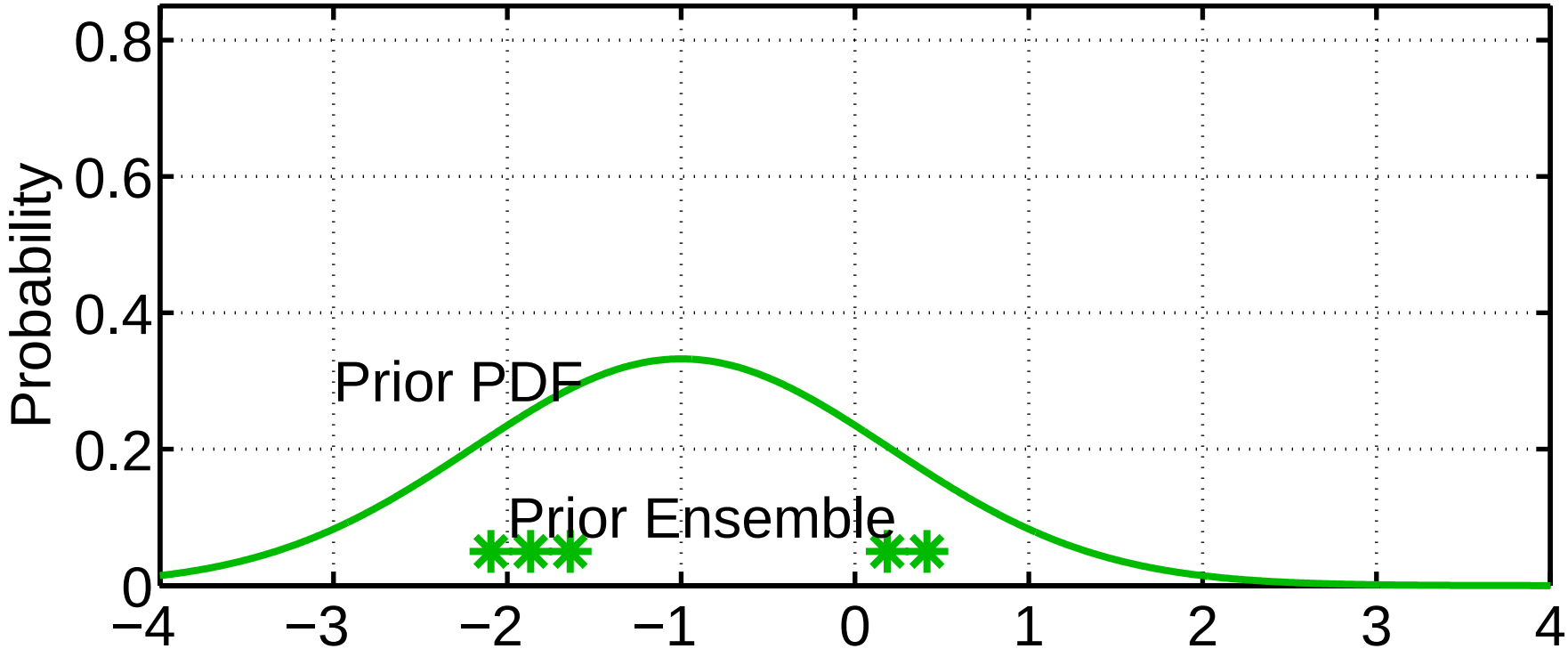


Can retain more correct information about non-Gaussian priors.

Can also be used for obs. likelihood term in product (not shown here).

Ensemble Filter Algorithms:

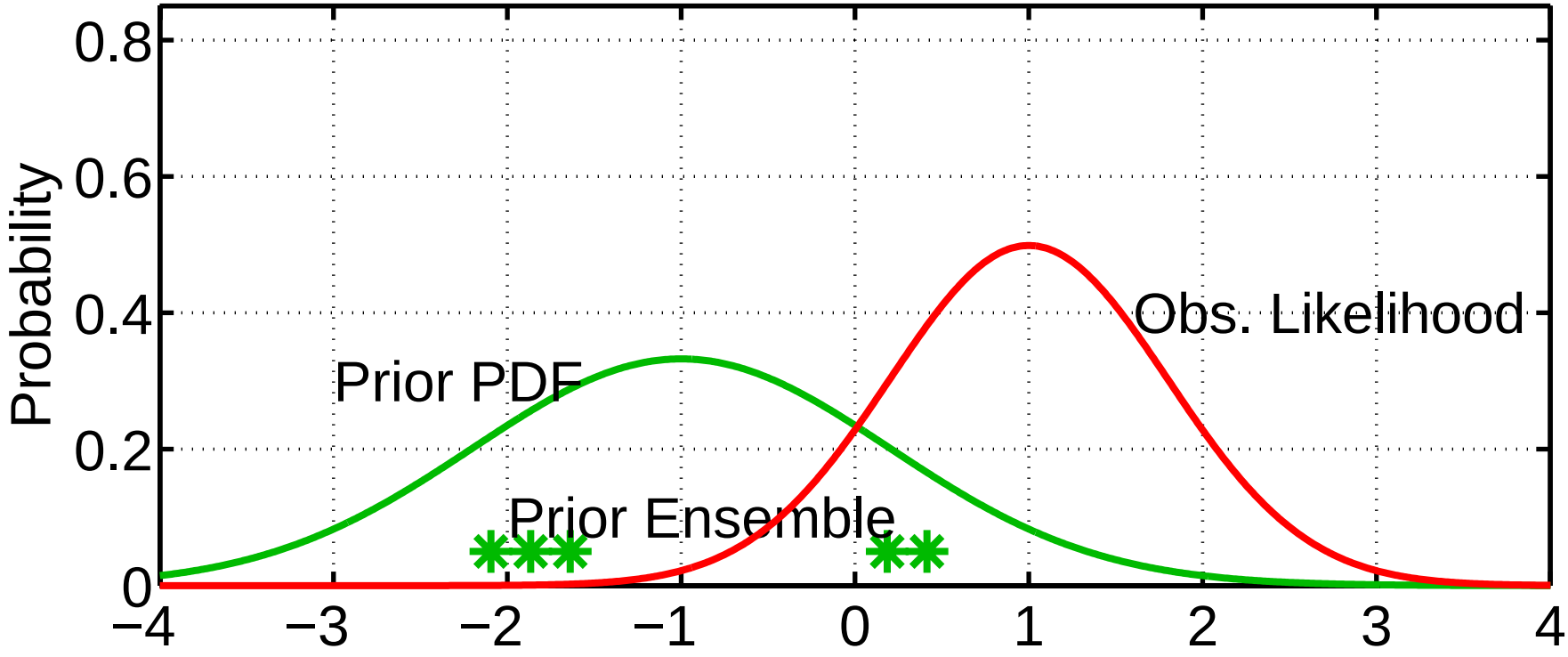
Ensemble Kernel filter (filter_kind=3 in assim_tools_nml).



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

Ensemble Filter Algorithms:

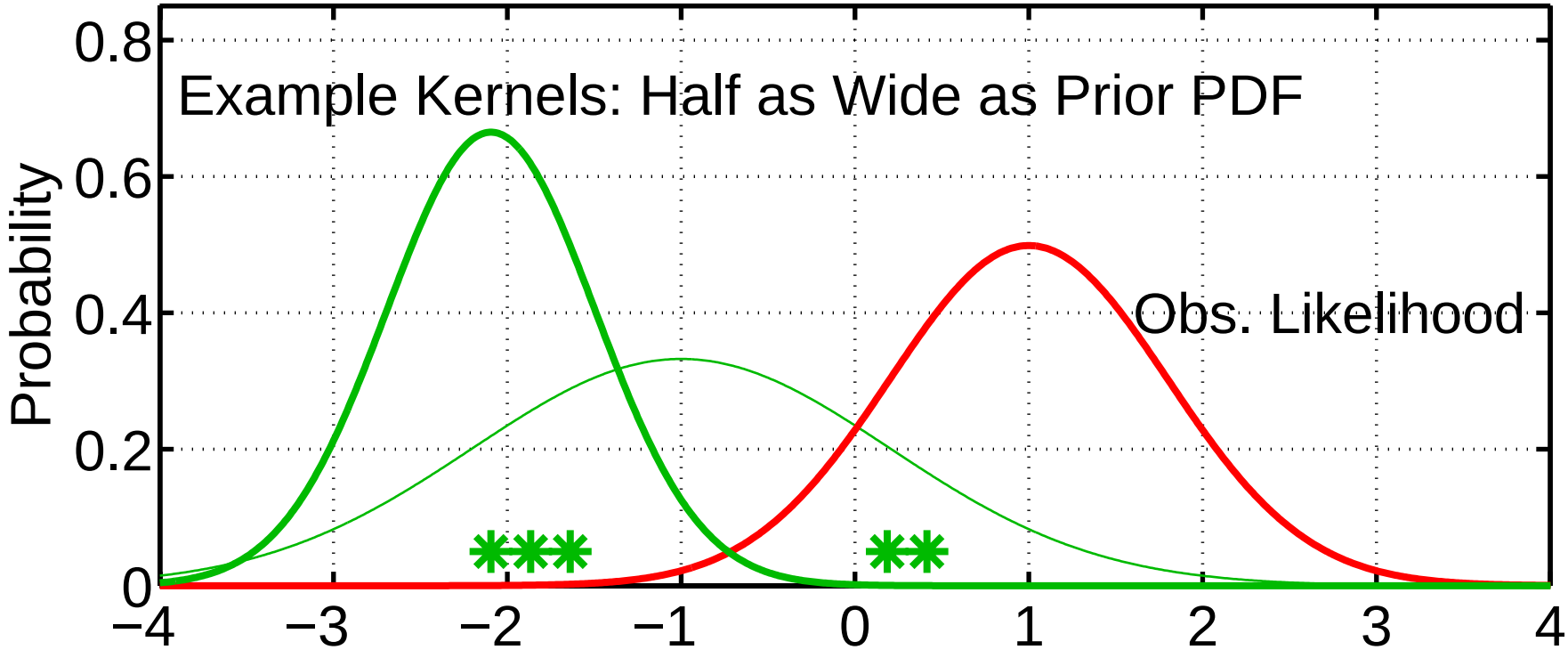
Ensemble Kernel filter (filter_kind=3 in assim_tools_nml).



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

Ensemble Filter Algorithms:

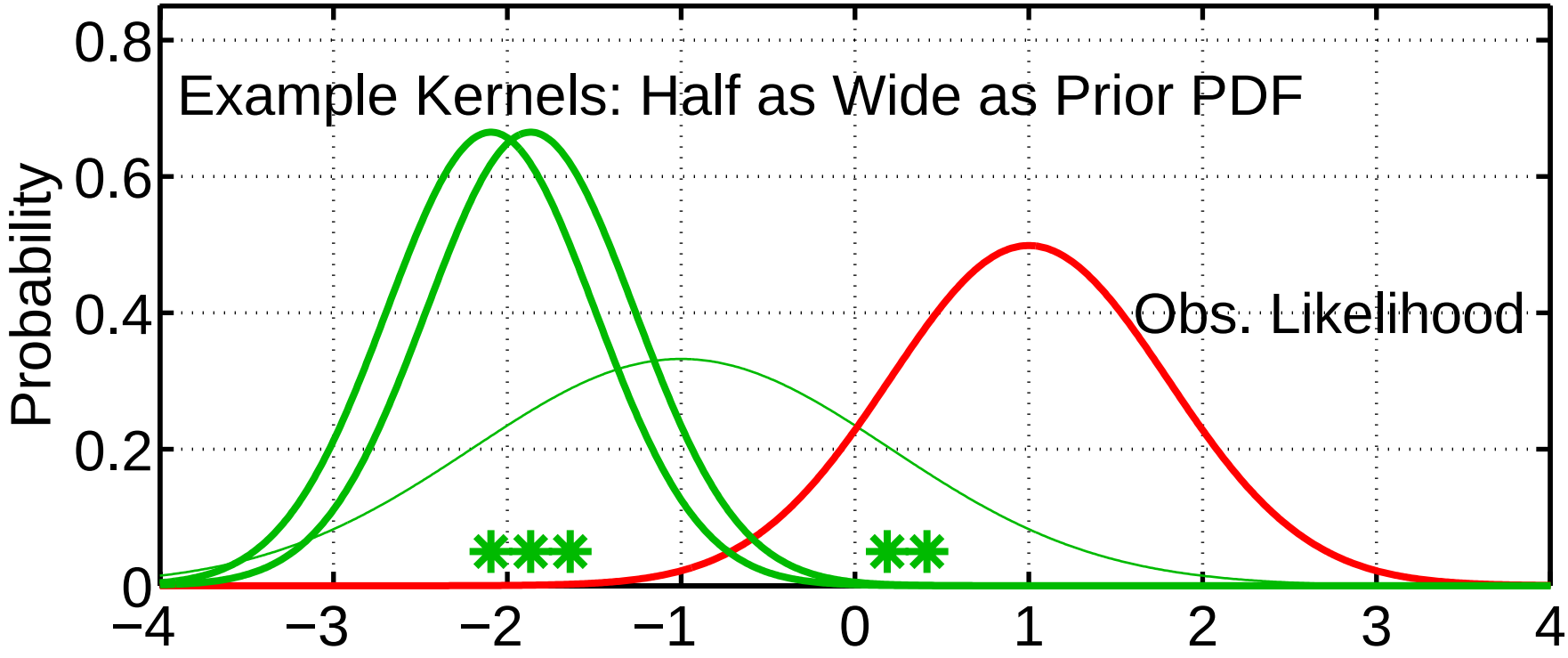
Ensemble Kernel filter (filter_kind=3 in assim_tools_nml).



Approximate prior as sum of Gaussians centered on each sample.

Ensemble Filter Algorithms:

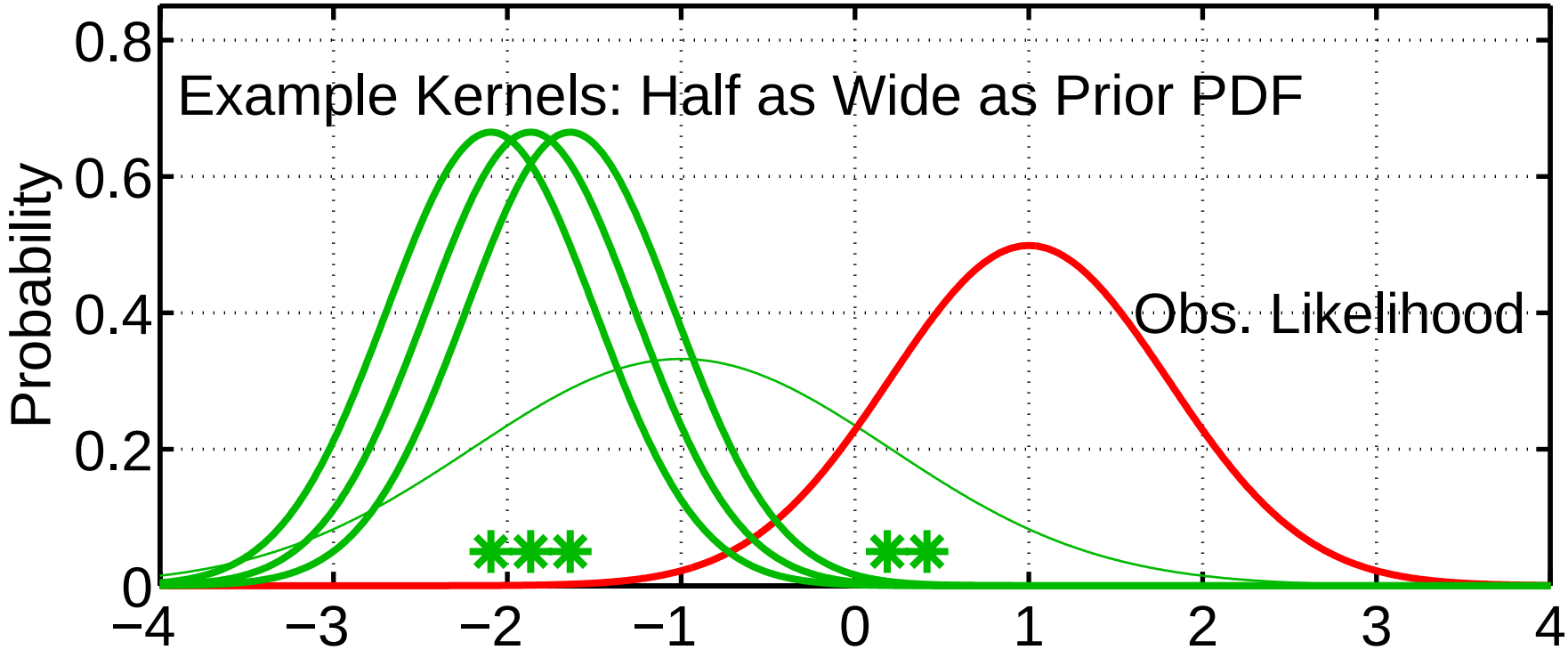
Ensemble Kernel filter (filter_kind=3 in assim_tools_nml).



Approximate prior as sum of Gaussians centered on each sample.

Ensemble Filter Algorithms:

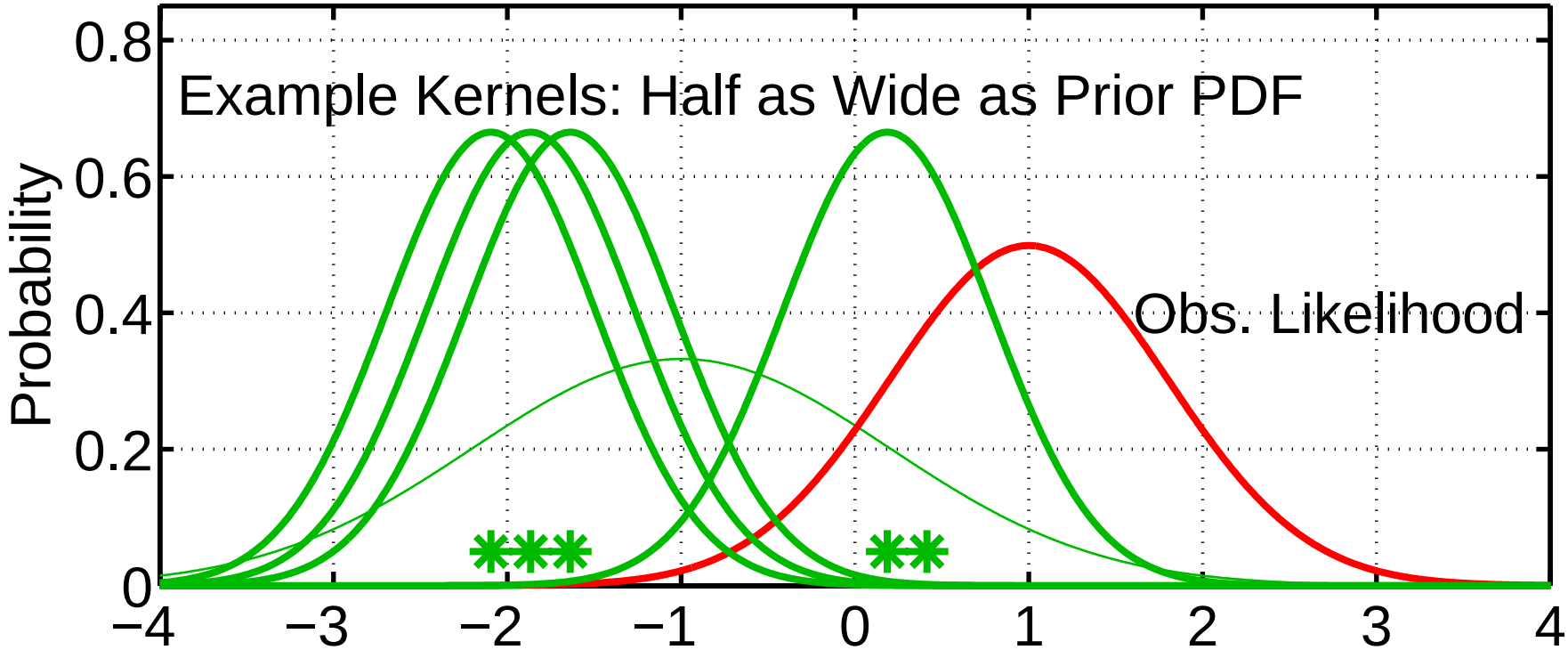
Ensemble Kernel filter (filter_kind=3 in assim_tools_nml).



Approximate prior as sum of Gaussians centered on each sample.

Ensemble Filter Algorithms:

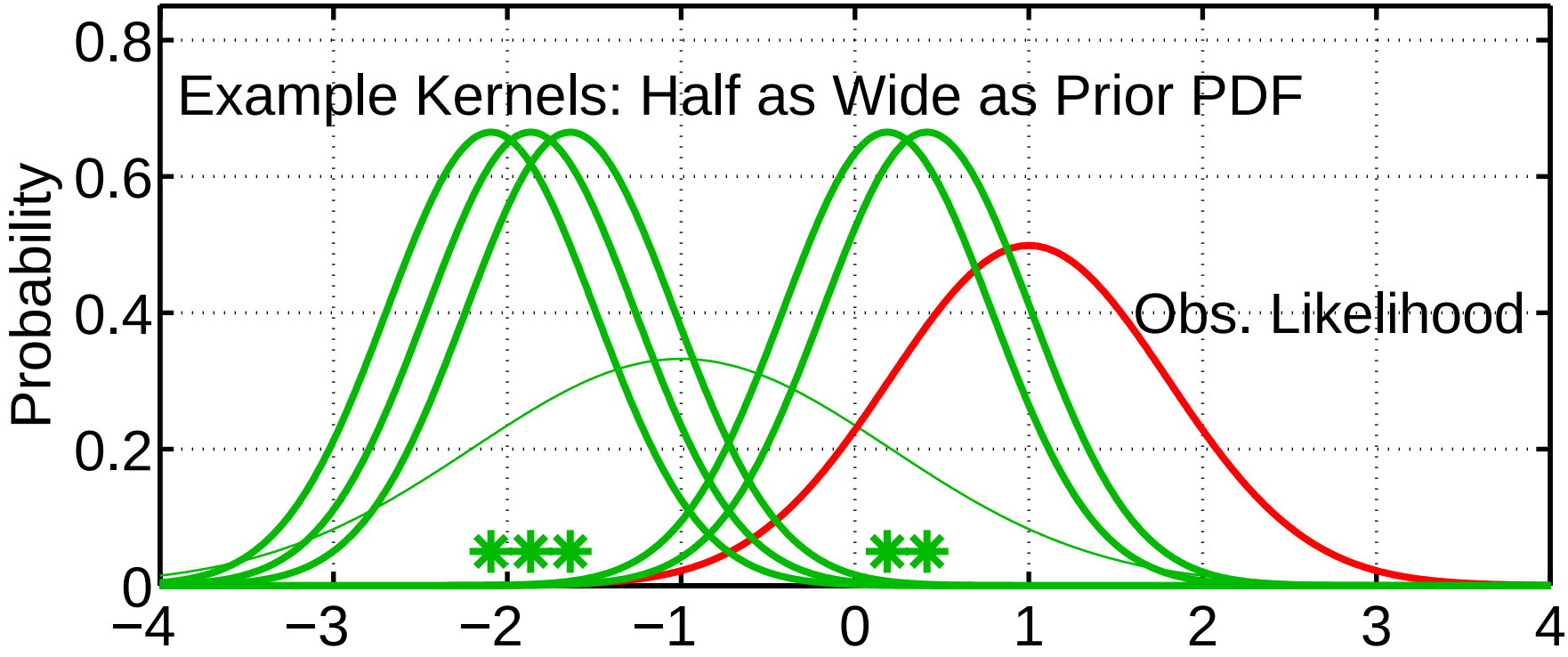
Ensemble Kernel filter (filter_kind=3 in assim_tools_nml).



Approximate prior as sum of Gaussians centered on each sample.

Ensemble Filter Algorithms:

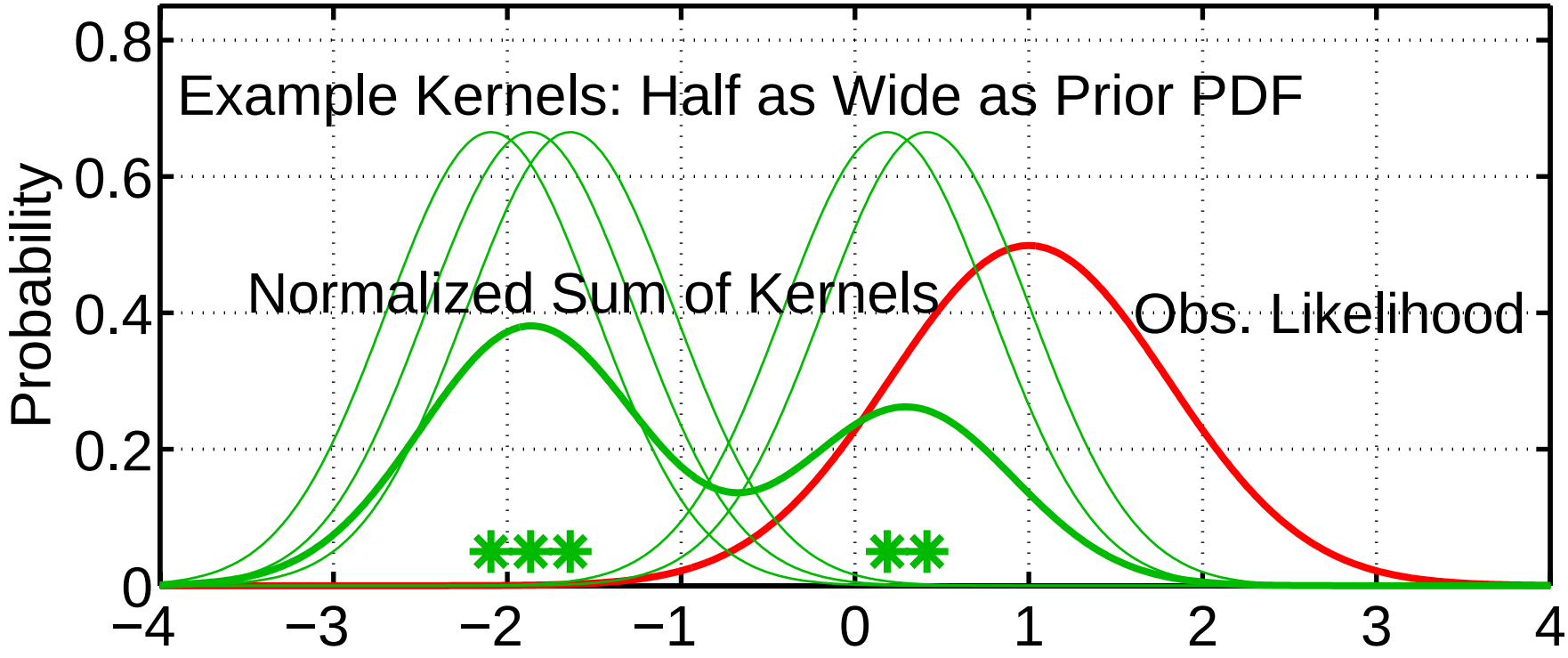
Ensemble Kernel filter (filter_kind=3 in assim_tools_nml).



Approximate prior as sum of Gaussians centered on each sample.

Ensemble Filter Algorithms:

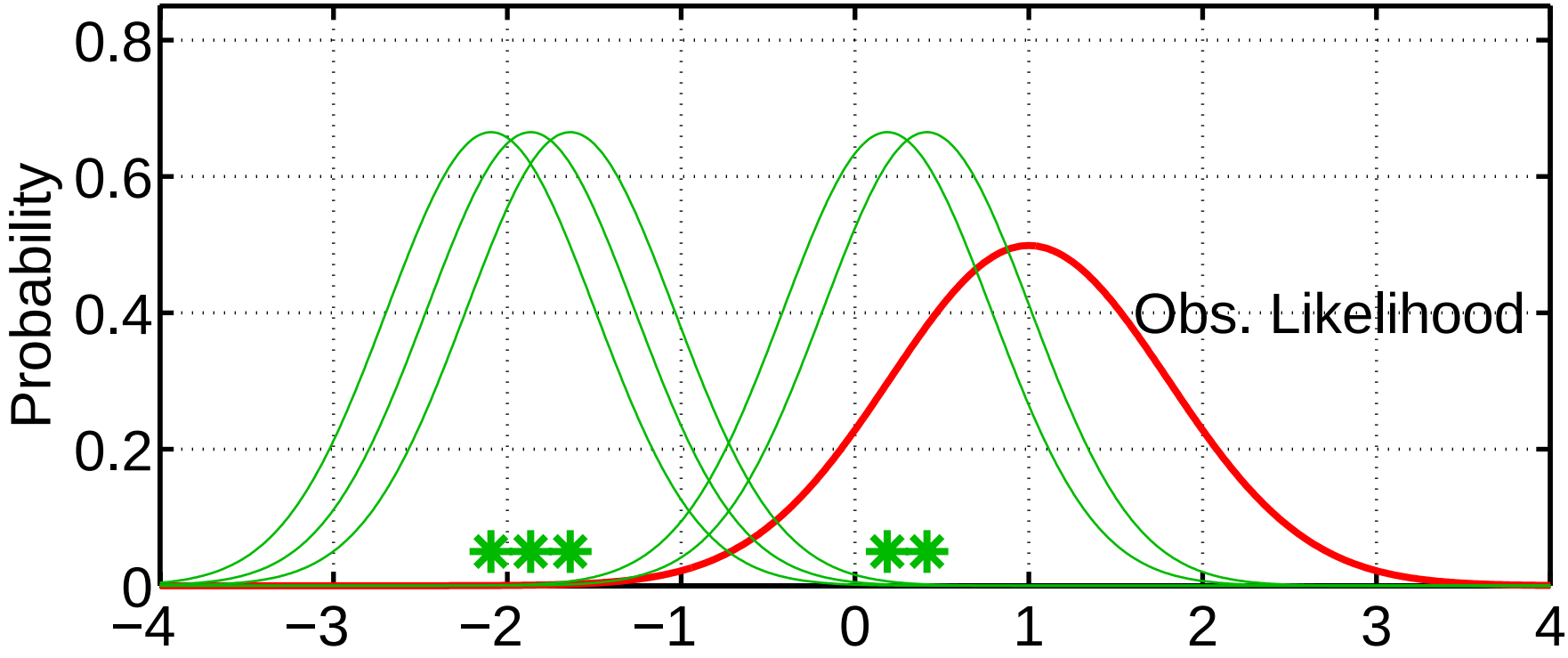
Ensemble Kernel filter (filter_kind=3 in assim_tools_nml).



Estimate of prior is normalized sum of all kernels.

Ensemble Filter Algorithms:

Ensemble Kernel filter (filter_kind=3 in assim_tools_nml).

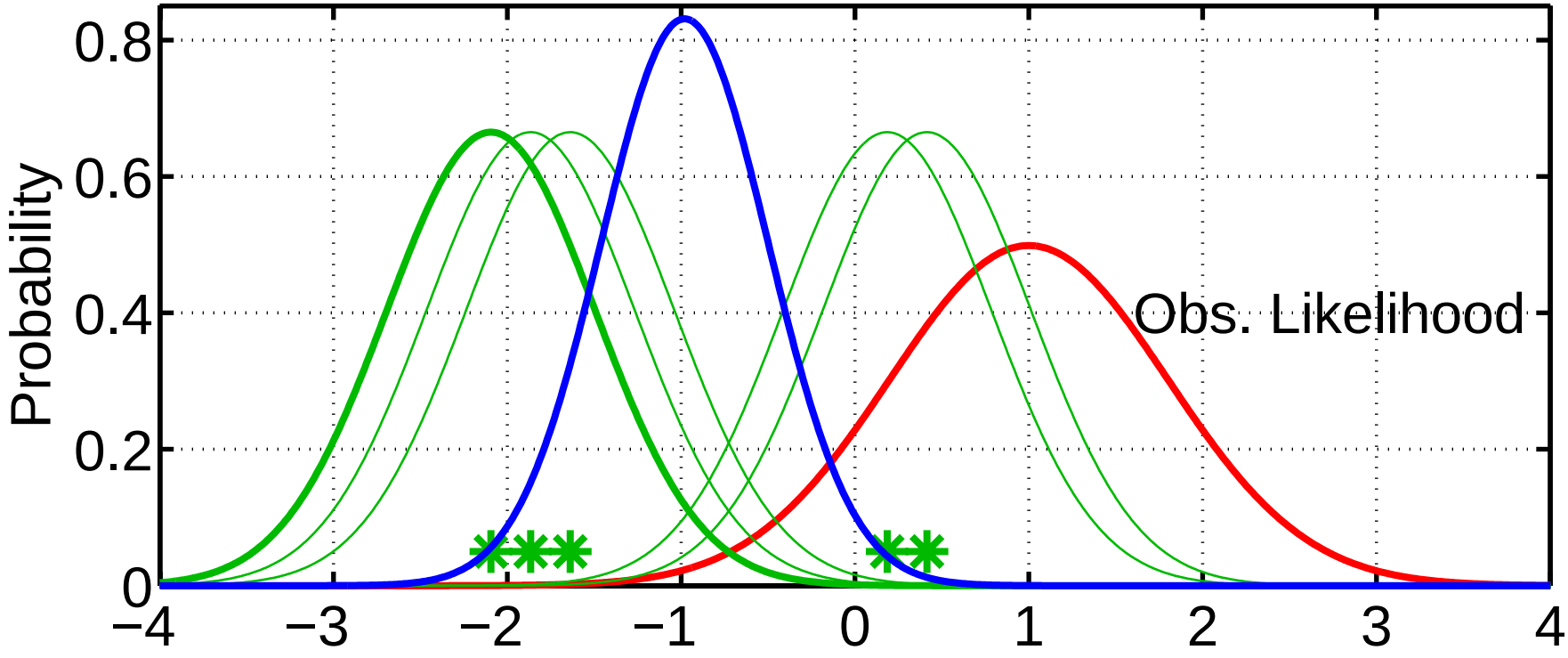


Apply distributive law to take product.

Product of sum is sum of products.

Ensemble Filter Algorithms:

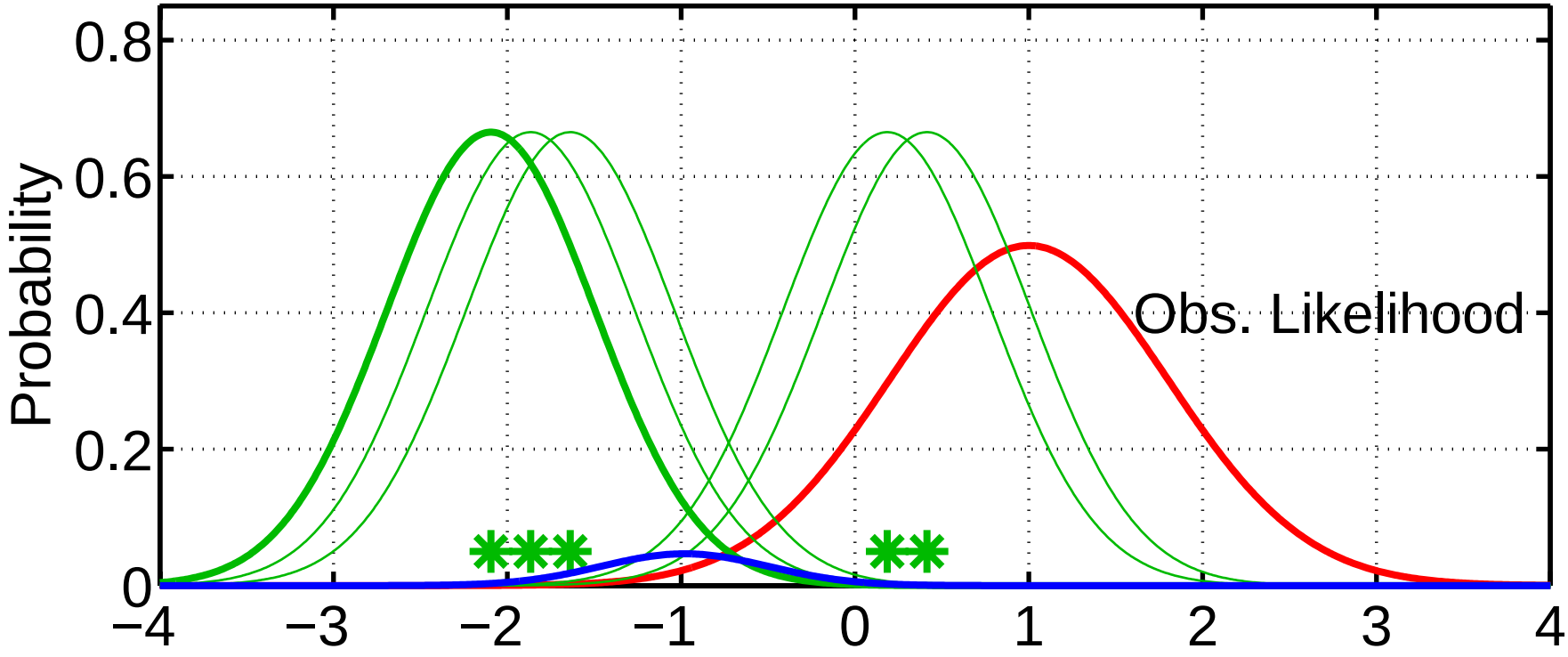
Ensemble Kernel filter (filter_kind=3 in assim_tools_nml).



Compute product of first kernel with Obs. Likelihood.

Ensemble Filter Algorithms:

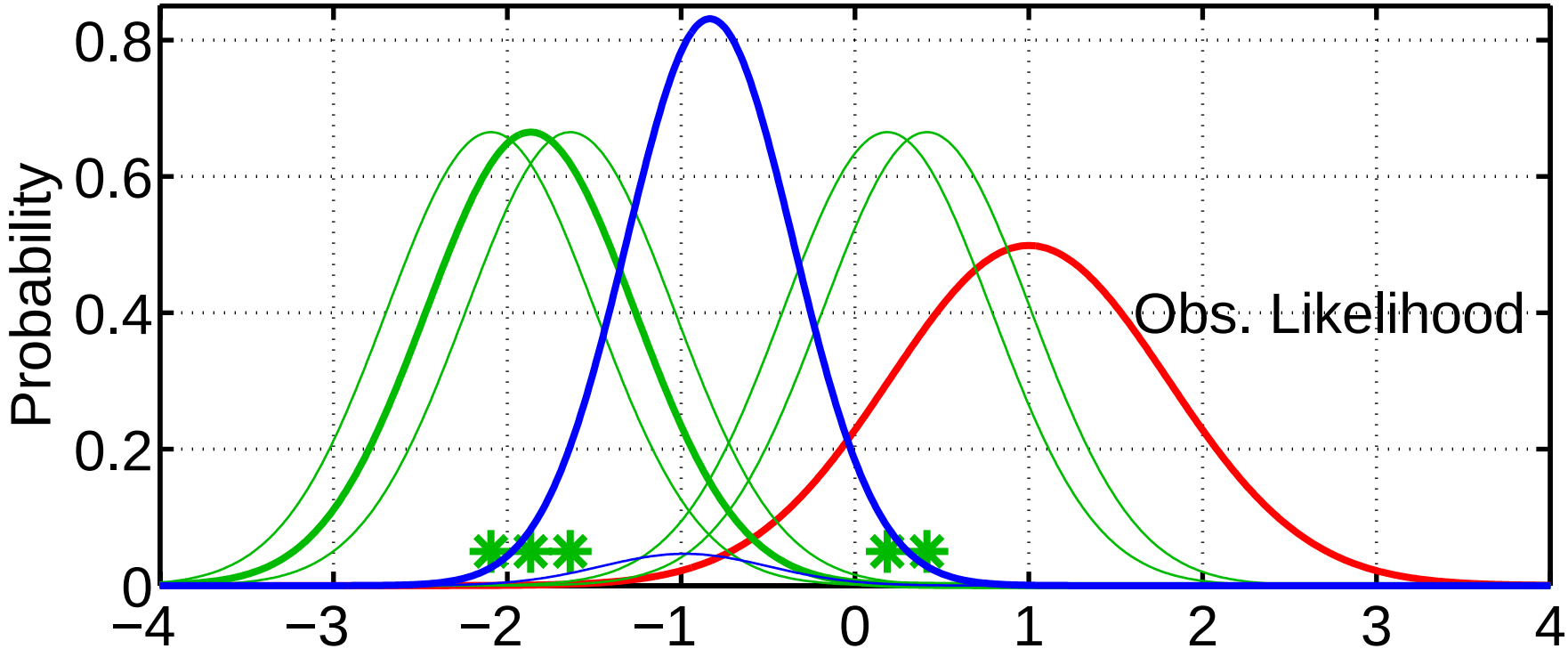
Ensemble Kernel filter (filter_kind=3 in assim_tools_nml).



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 (filter_kind=3 in assim_tools_nml).

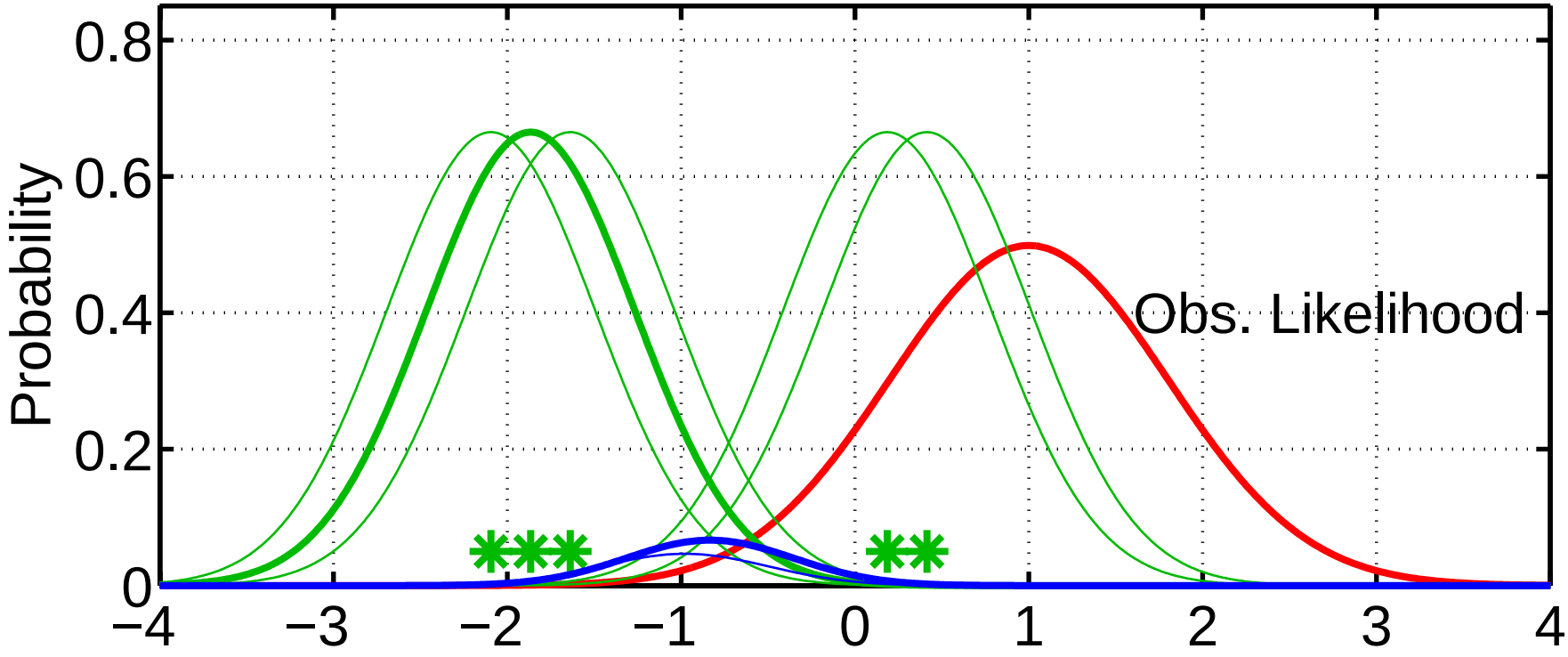


Continue to take products for each kernel in turn.

More distant kernels have small impact on posterior.

Ensemble Filter Algorithms:

Ensemble Kernel filter (filter_kind=3 in assim_tools_nml).

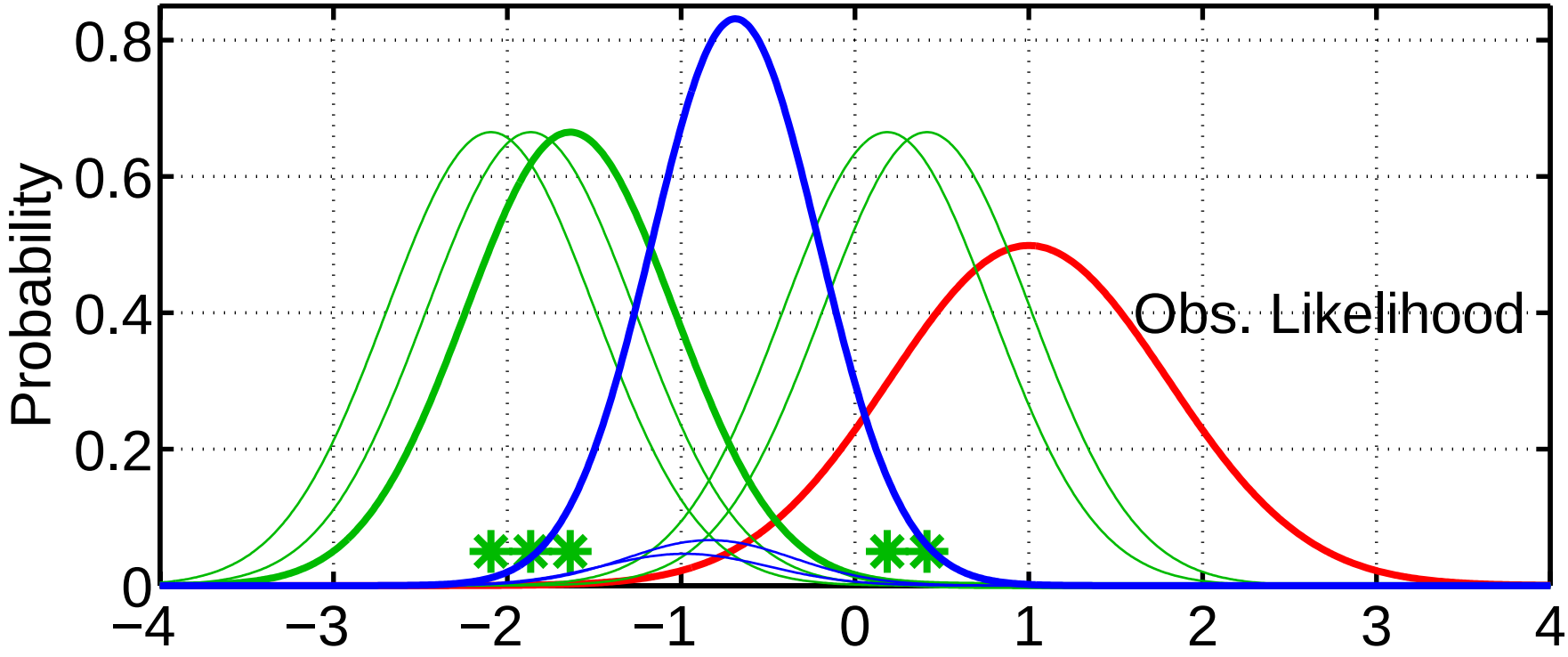


Continue to take products for each kernel in turn.

More distant kernels have small impact on posterior.

Ensemble Filter Algorithms:

Ensemble Kernel filter (filter_kind=3 in assim_tools_nml).

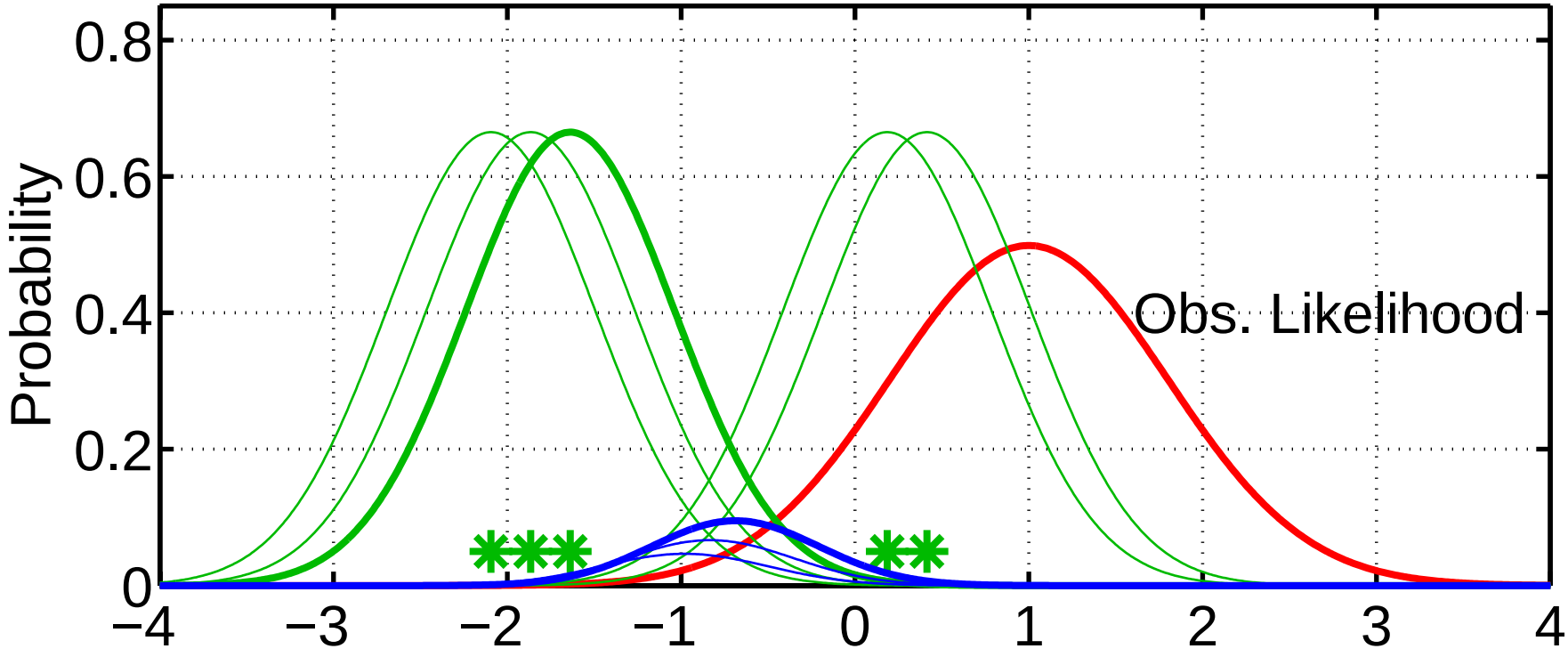


Continue to take products for each kernel in turn.

More distant kernels have small impact on posterior.

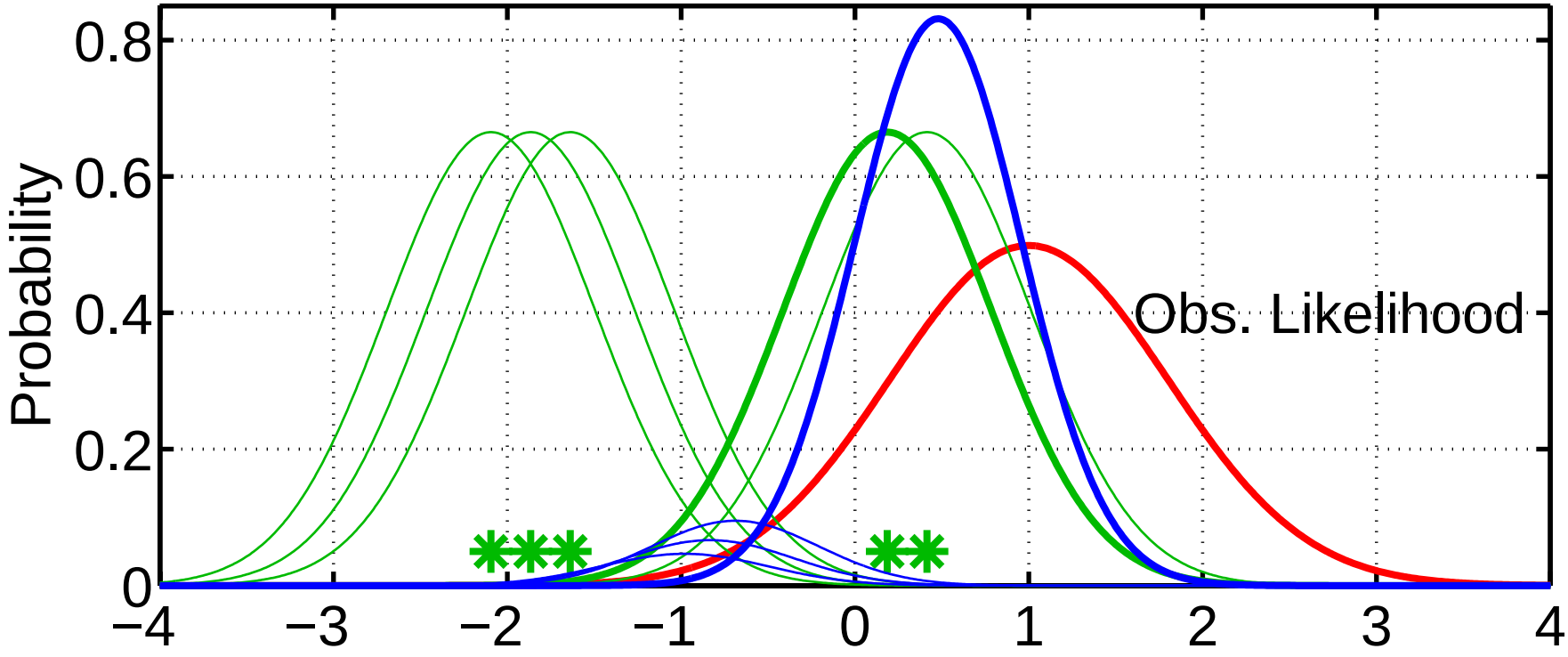
Ensemble Filter Algorithms:

Ensemble Kernel filter (filter_kind=3 in assim_tools_nml).



Ensemble Filter Algorithms:

Ensemble Kernel filter (filter_kind=3 in assim_tools_nml).

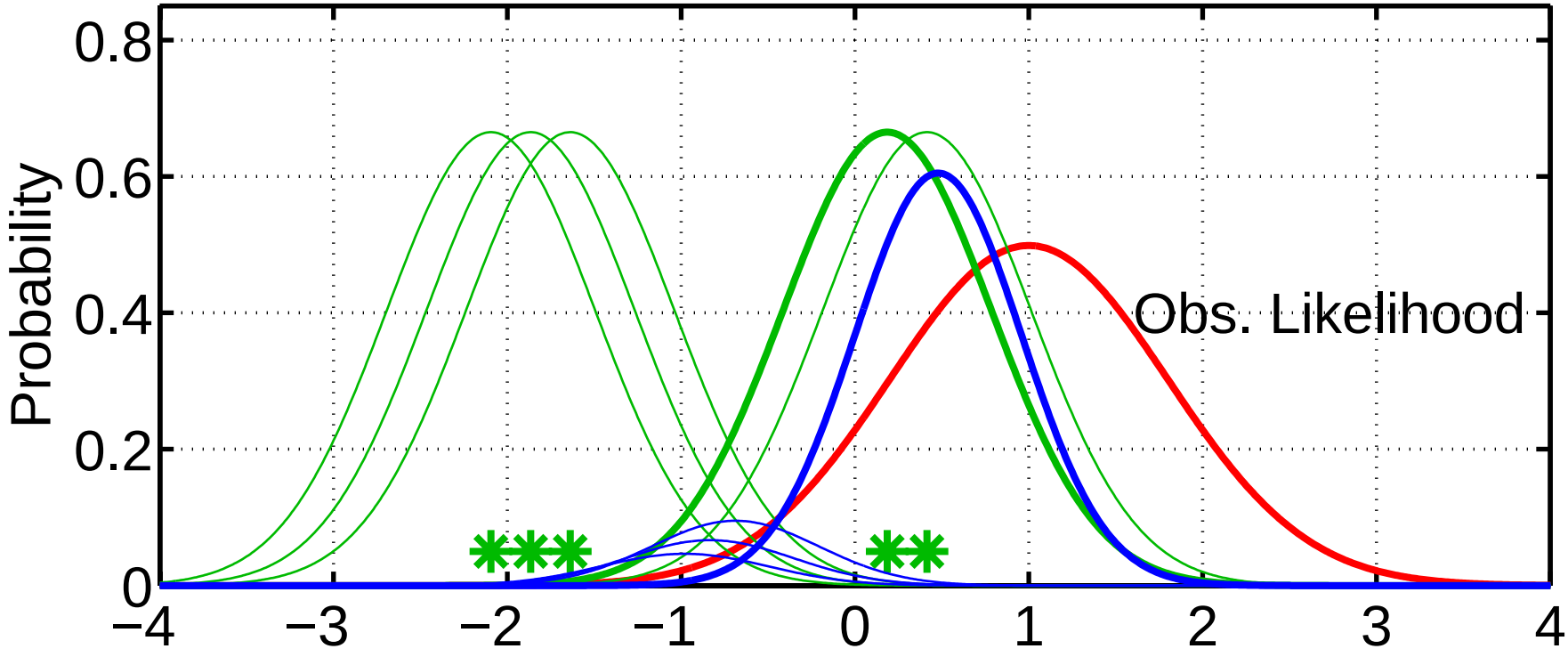


Continue to take products for each kernel in turn.

Closer kernels dominate posterior.

Ensemble Filter Algorithms:

Ensemble Kernel filter (filter_kind=3 in assim_tools_nml).

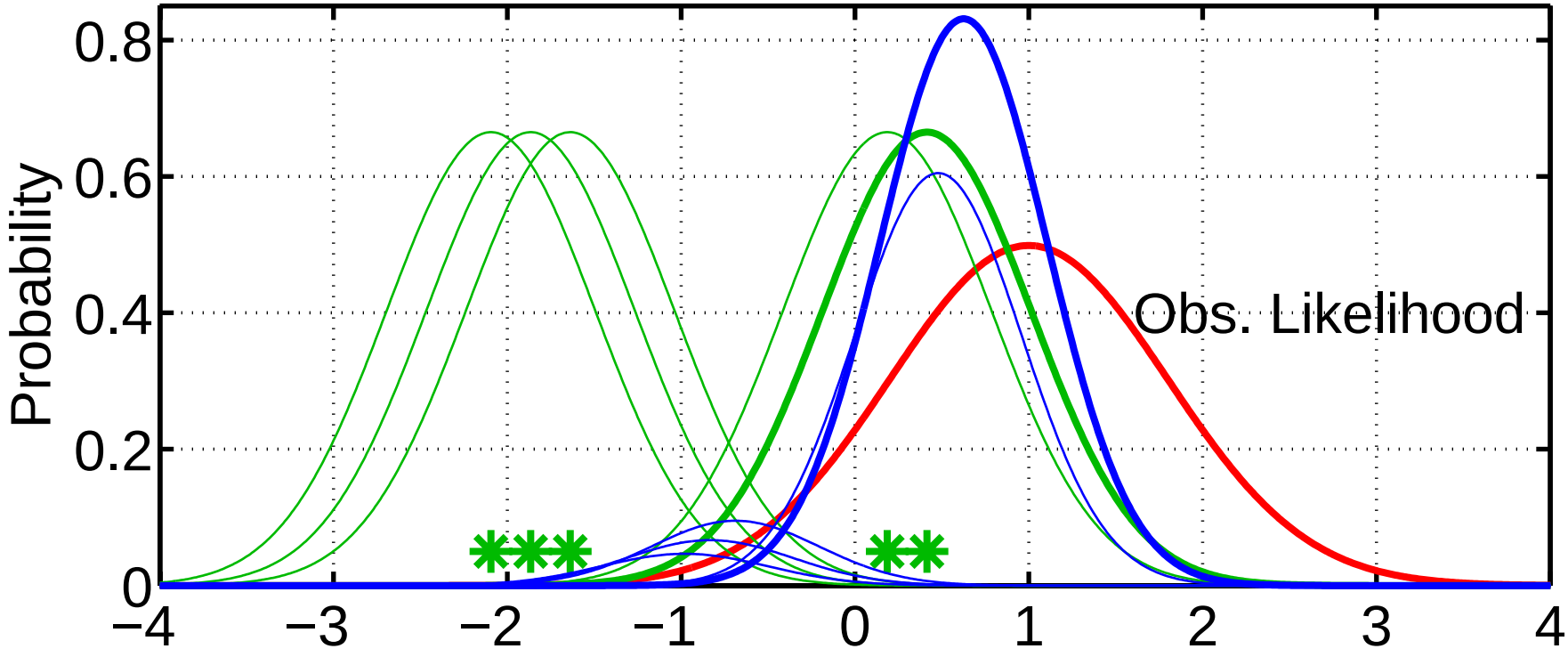


Continue to take products for each kernel in turn.

Closer kernels dominate posterior.

Ensemble Filter Algorithms:

Ensemble Kernel filter (filter_kind=3 in assim_tools_nml).

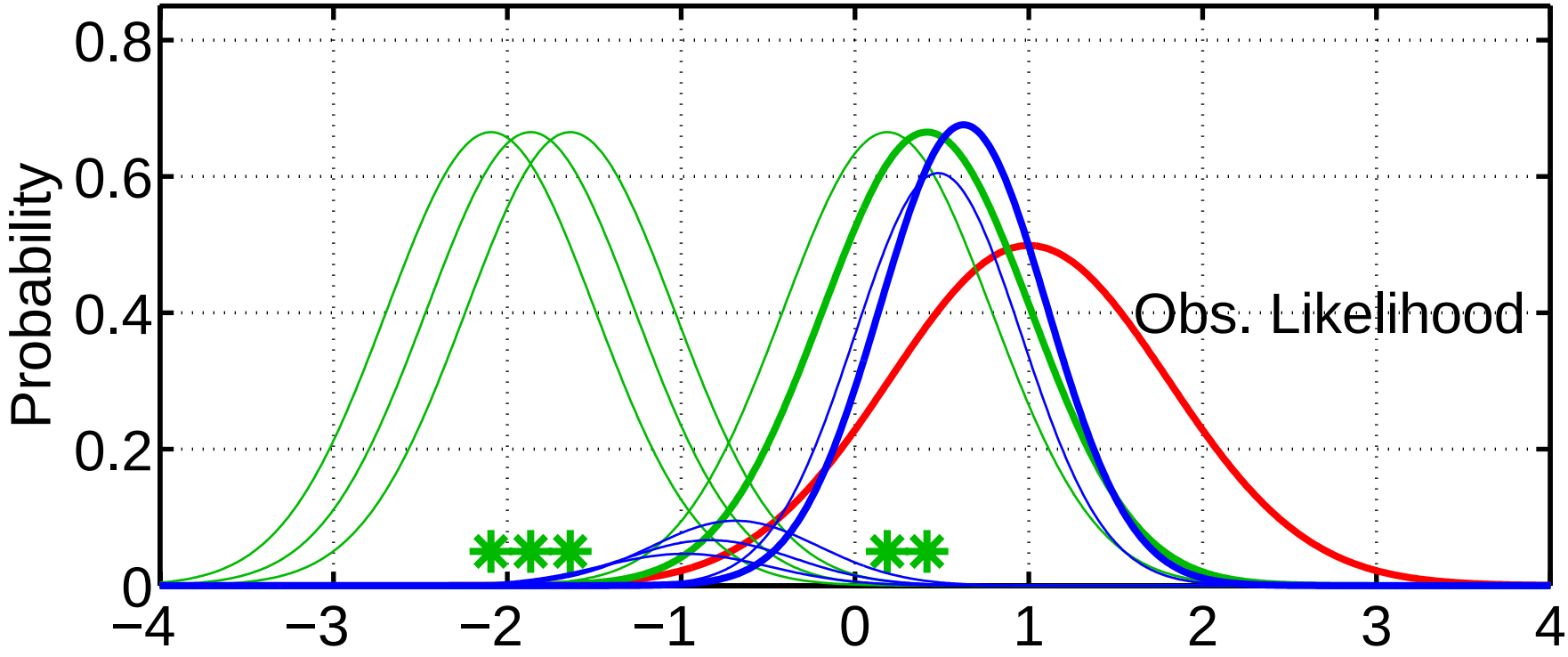


Continue to take products for each kernel in turn.

Closer kernels dominate posterior.

Ensemble Filter Algorithms:

Ensemble Kernel filter (filter_kind=3 in assim_tools_nml).

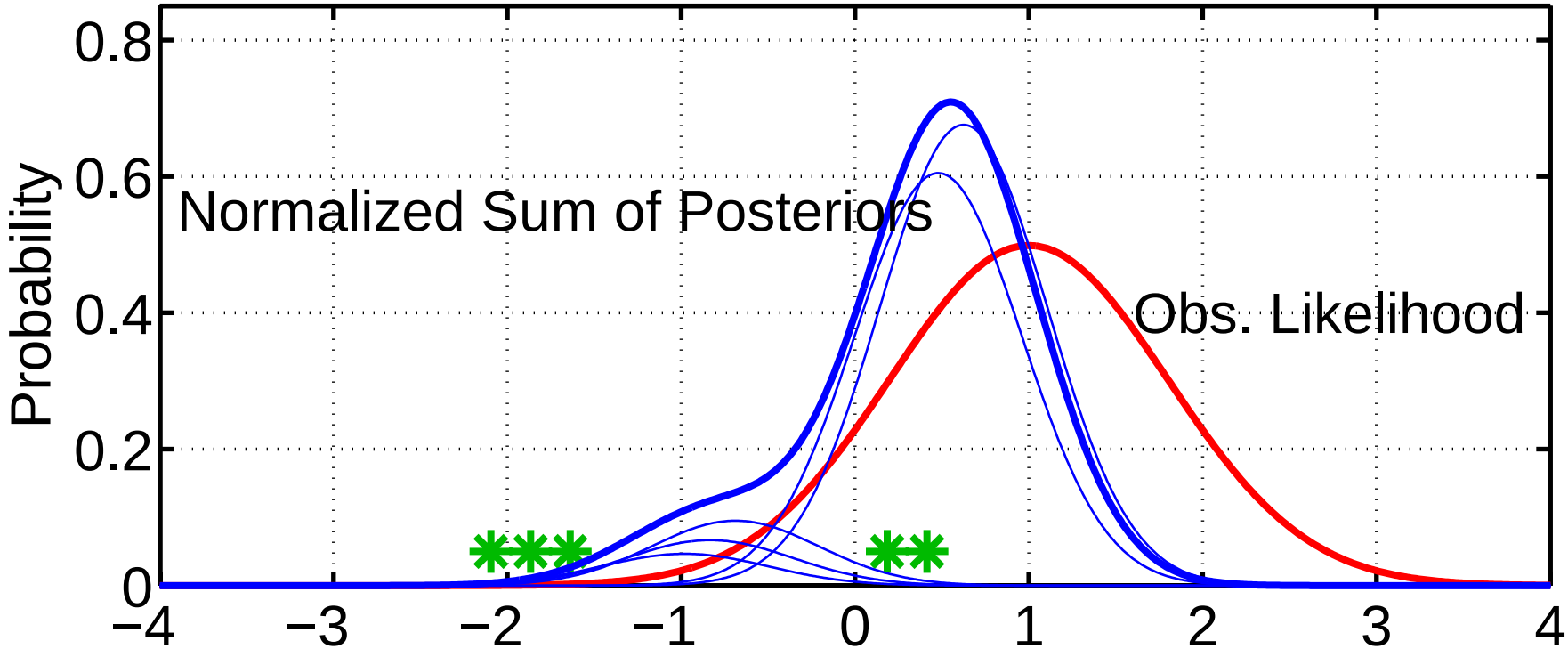


Continue to take products for each kernel in turn.

Closer kernels dominate posterior.

Ensemble Filter Algorithms:

Ensemble Kernel filter (filter_kind=3 in assim_tools_nml).

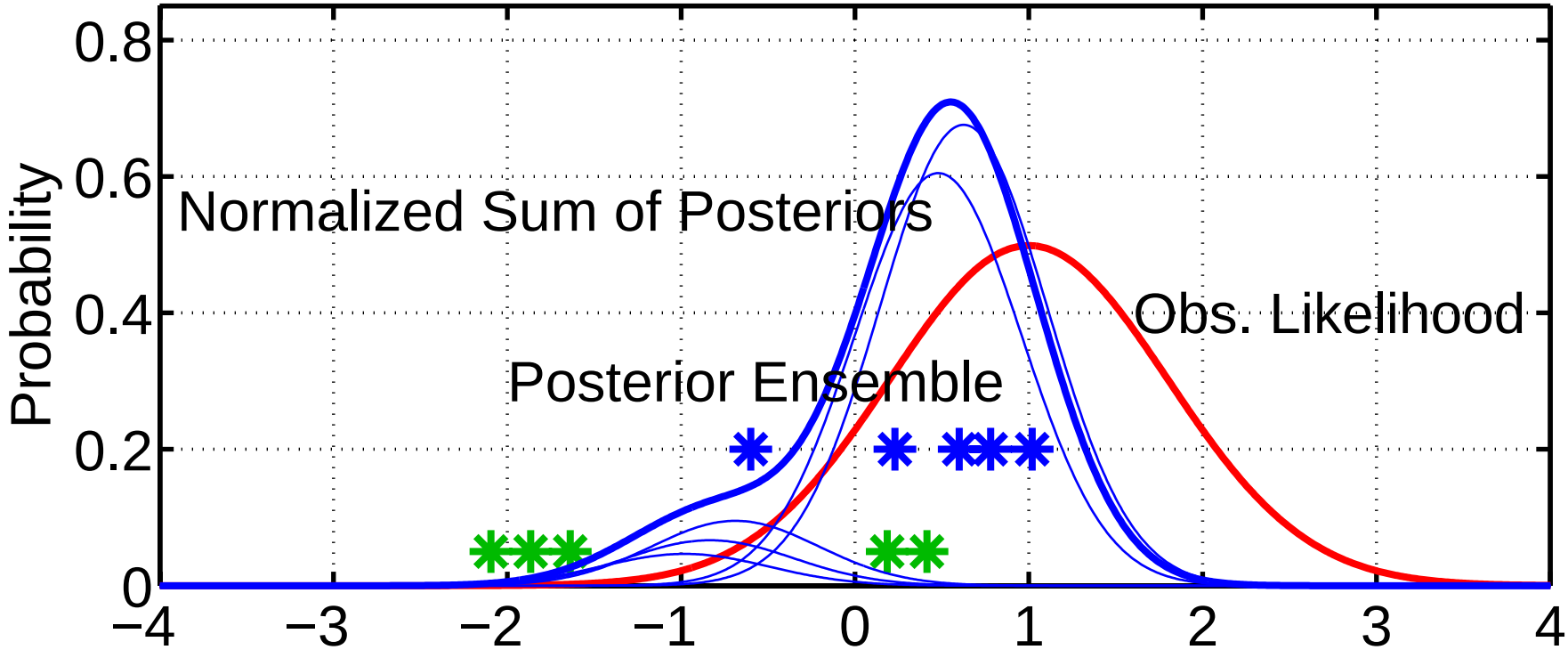


Final posterior is weight-normalized sum of kernel products.

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

Ensemble Filter Algorithms:

Ensemble Kernel filter (filter_kind=3 in assim_tools_nml).



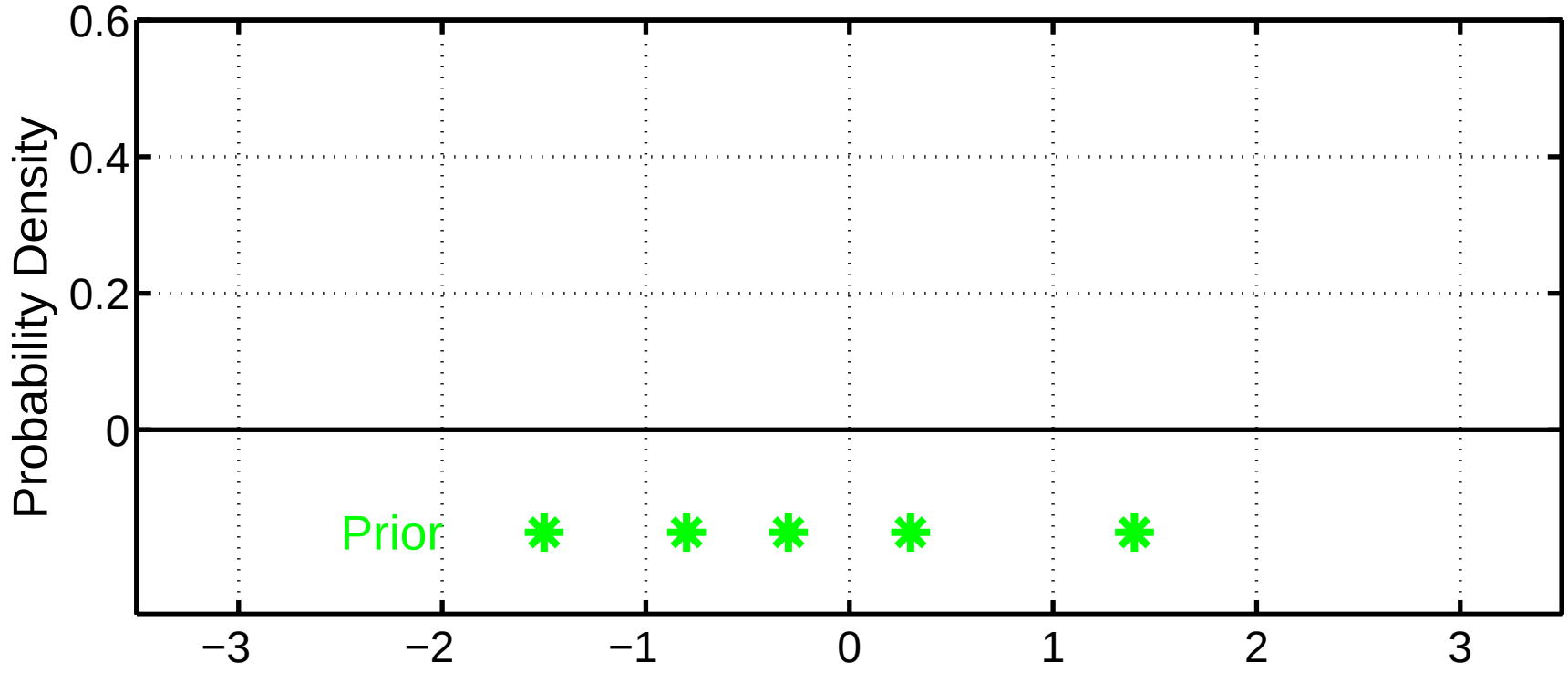
Forming sample of the posterior can be problematic.

Random sample is simple.

Deterministic sampling is much more tricky here (few results available)

Ensemble Filter Algorithms:

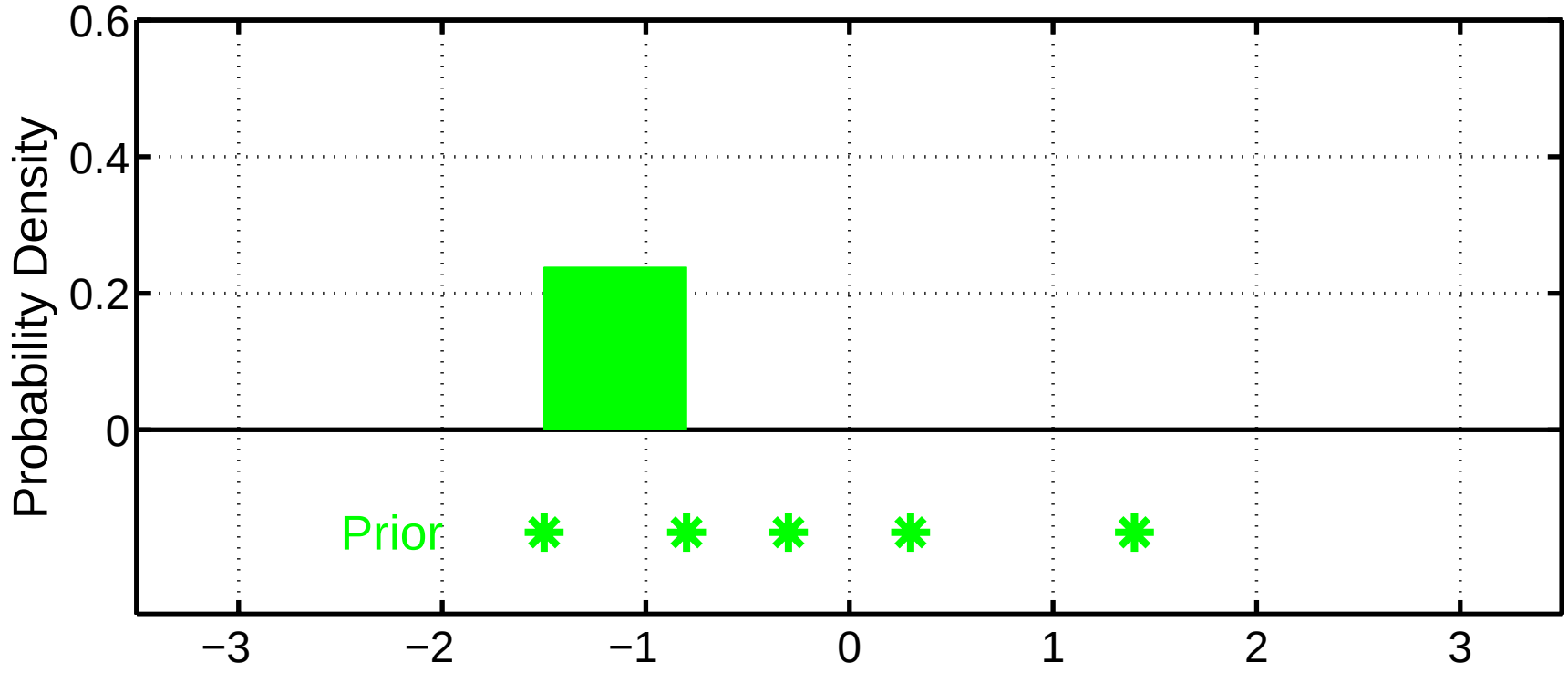
Boxcar Kernel filter (filter_kind=7 in assim_tools_nml).



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

Ensemble Filter Algorithms:

Boxcar Kernel filter (filter_kind=7 in assim_tools_nml).

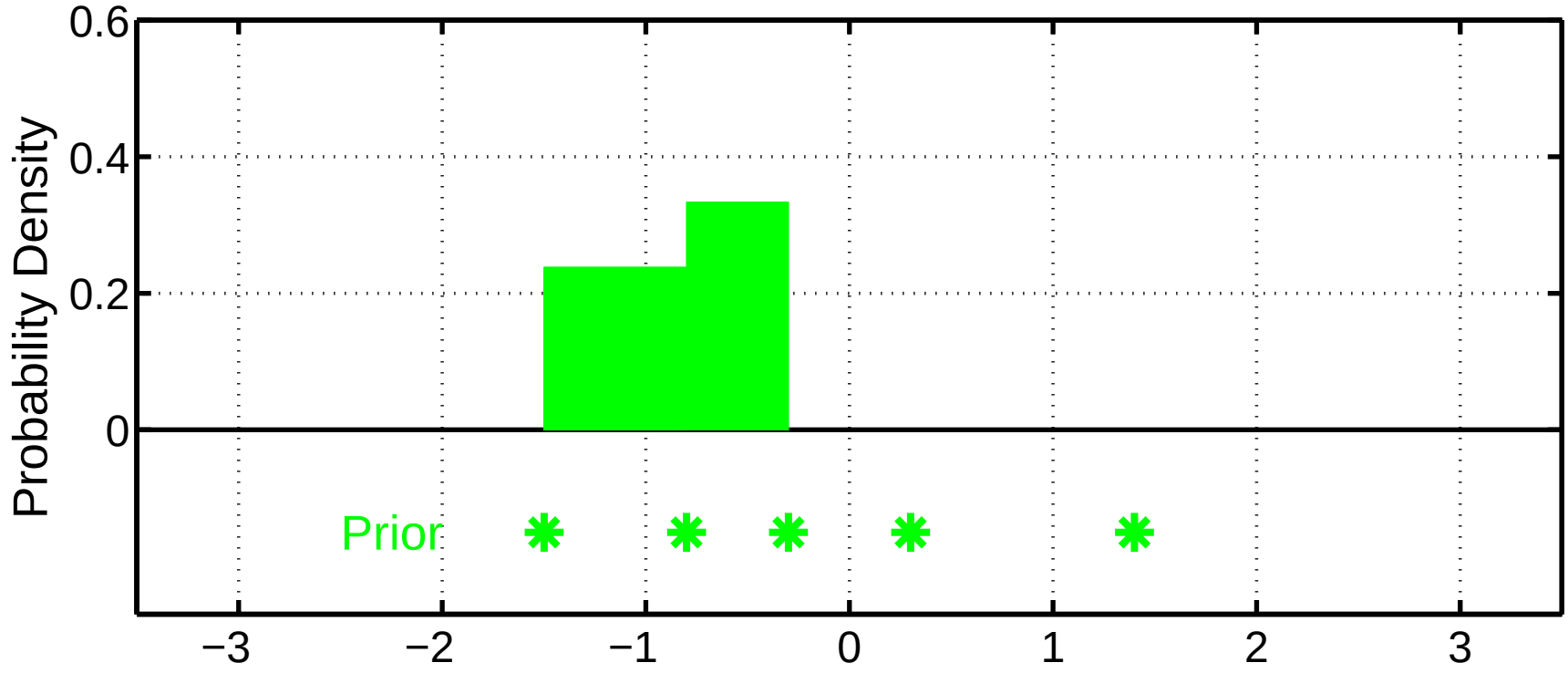


Define a continuous prior distribution:

1. Put 1/(ens_size + 1) mass uniformly between each ensemble pair.

Ensemble Filter Algorithms:

Boxcar Kernel filter (filter_kind=7 in assim_tools_nml).

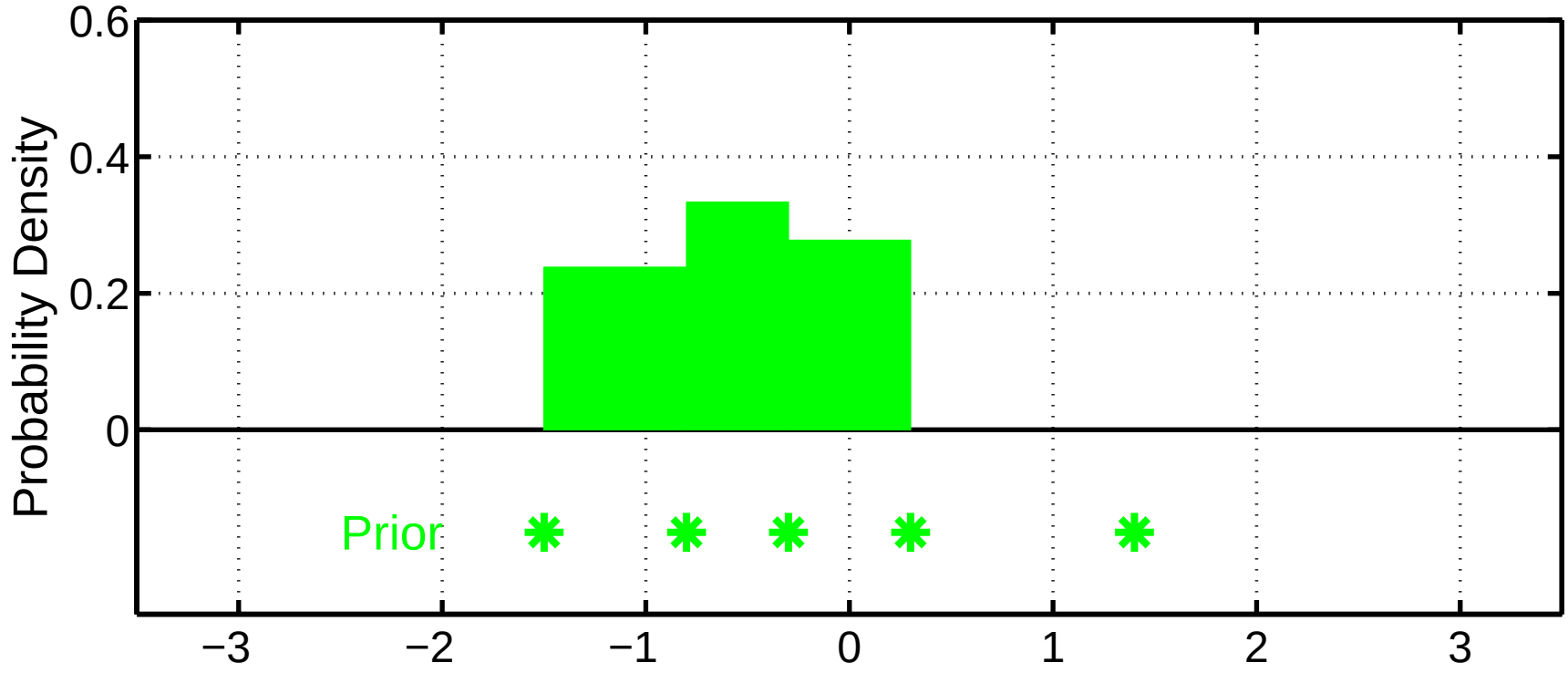


Define a continuous prior distribution:

1. Put 1/(ens_size + 1) mass uniformly between each ensemble pair.

Ensemble Filter Algorithms:

Boxcar Kernel filter (filter_kind=7 in assim_tools_nml).

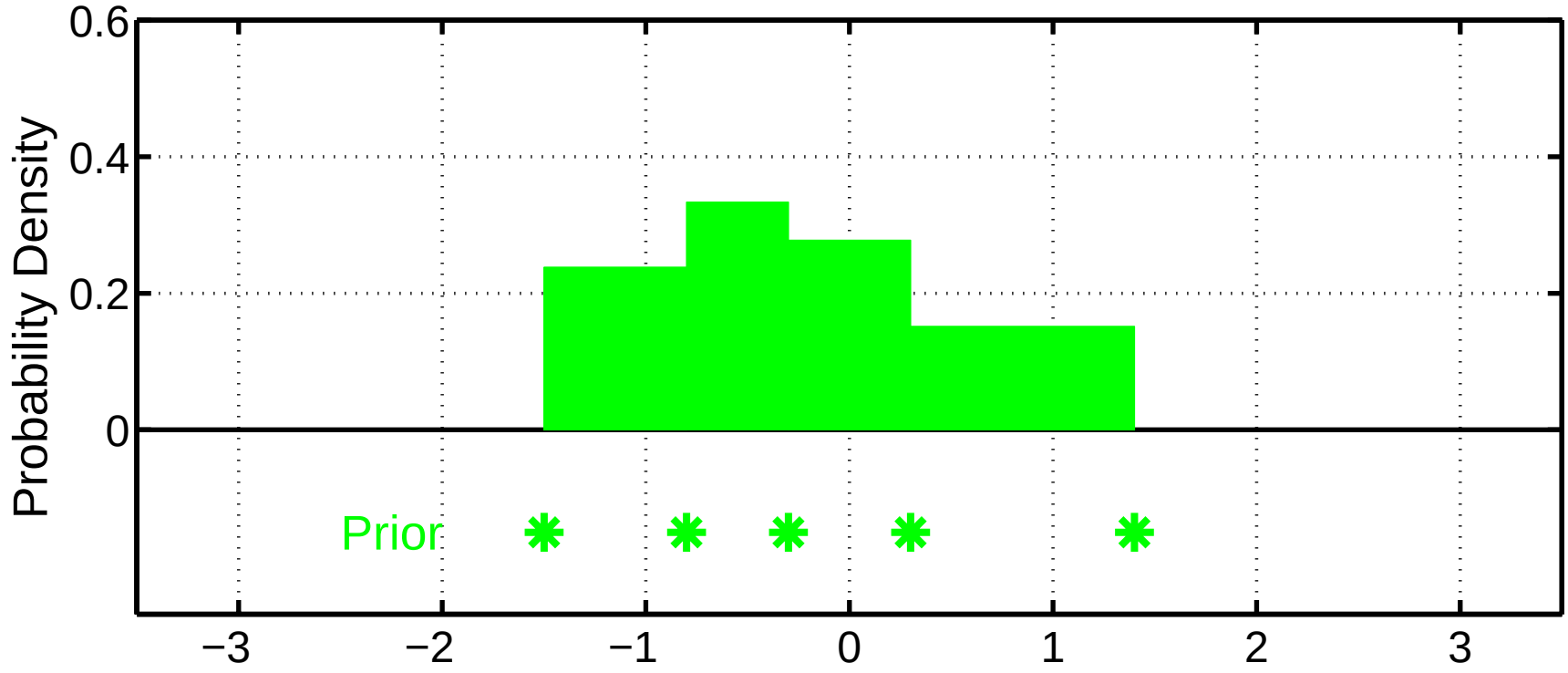


Define a continuous prior distribution:

1. Put 1/(ens_size + 1) mass uniformly between each ensemble pair.

Ensemble Filter Algorithms:

Boxcar Kernel filter (filter_kind=7 in assim_tools_nml).

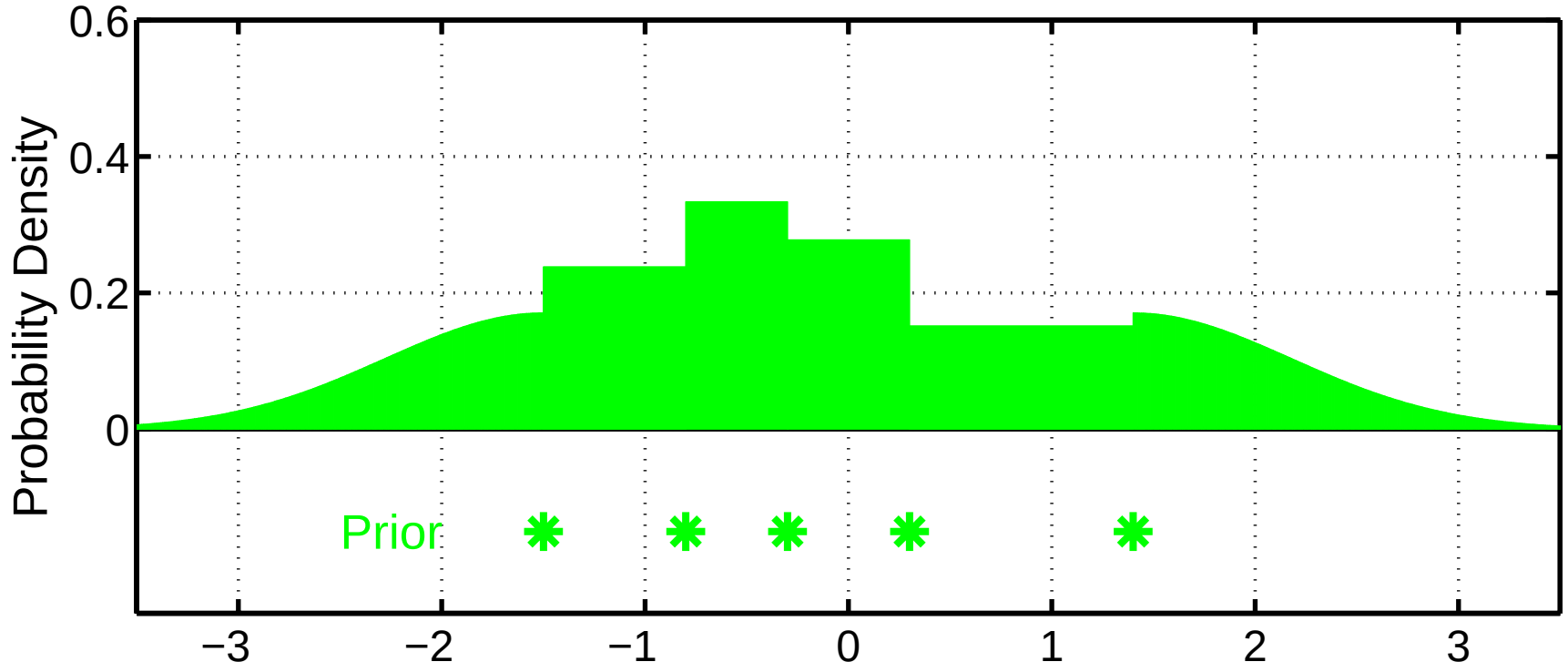


Define a continuous prior distribution:

1. Put 1/(ens_size + 1) mass uniformly between each ensemble pair.

Ensemble Filter Algorithms:

Boxcar Kernel filter (filter_kind=7 in assim_tools_nml).



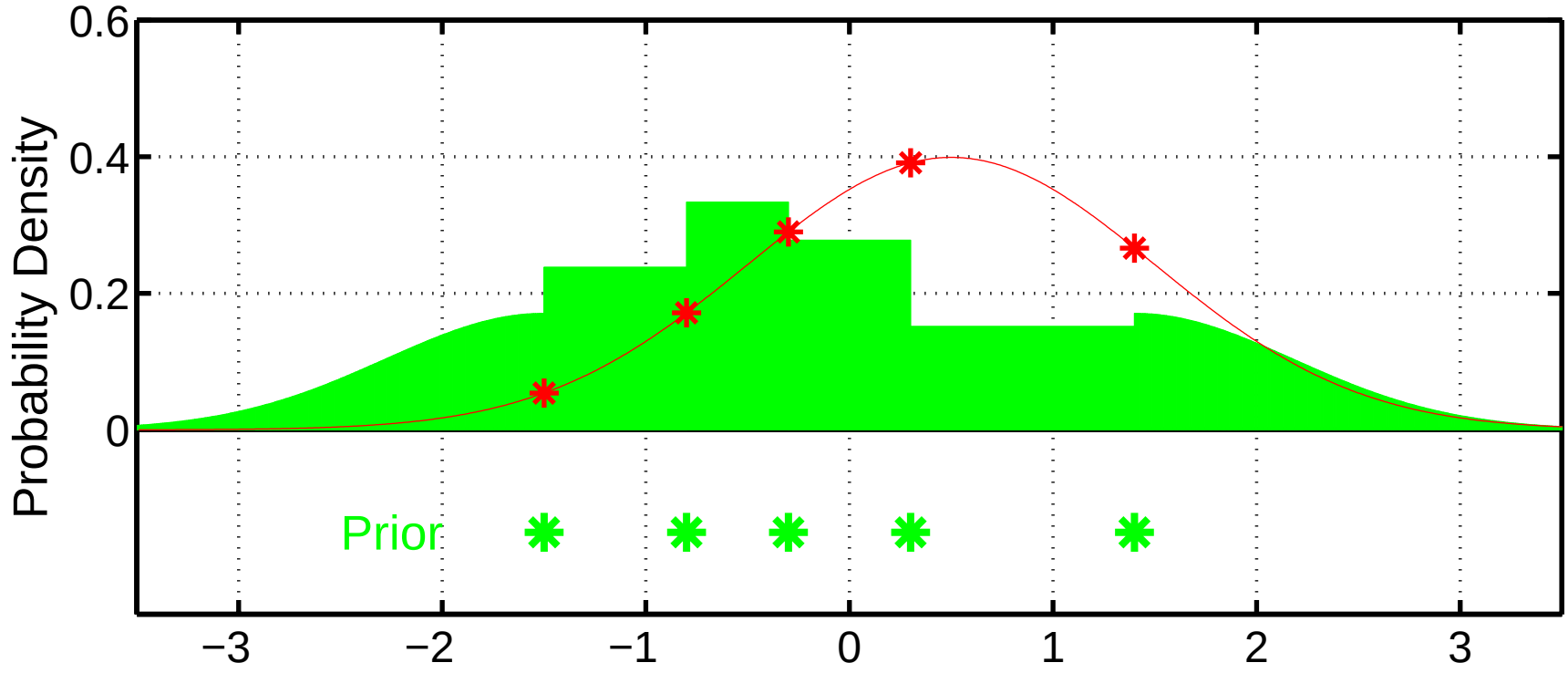
Define a continuous prior distribution:

1. Put 1/(ens_size + 1) mass uniformly between each ensemble pair.
2. Add N(Outermost Ensemble Member, $\sigma_{ens}/2$) kernels on edges.

This avoids filter divergence, even in presence of model bias.

Ensemble Filter Algorithms:

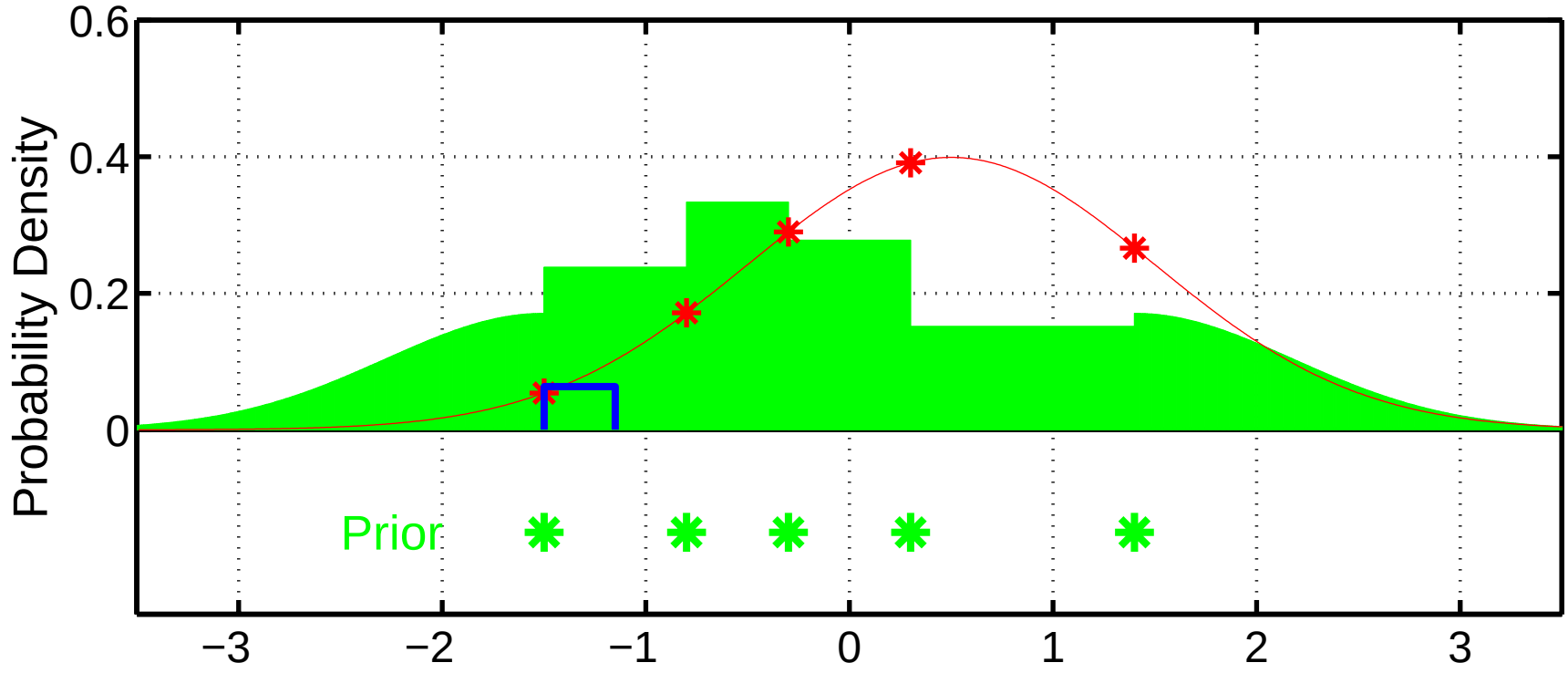
Boxcar Kernel filter (filter_kind=7 in assim_tools_nml).



Evaluate the likelihood at each ensemble member.
Could be done for an arbitrary non-gaussian likelihood.

Ensemble Filter Algorithms:

Boxcar Kernel filter (filter_kind=7 in assim_tools_nml).

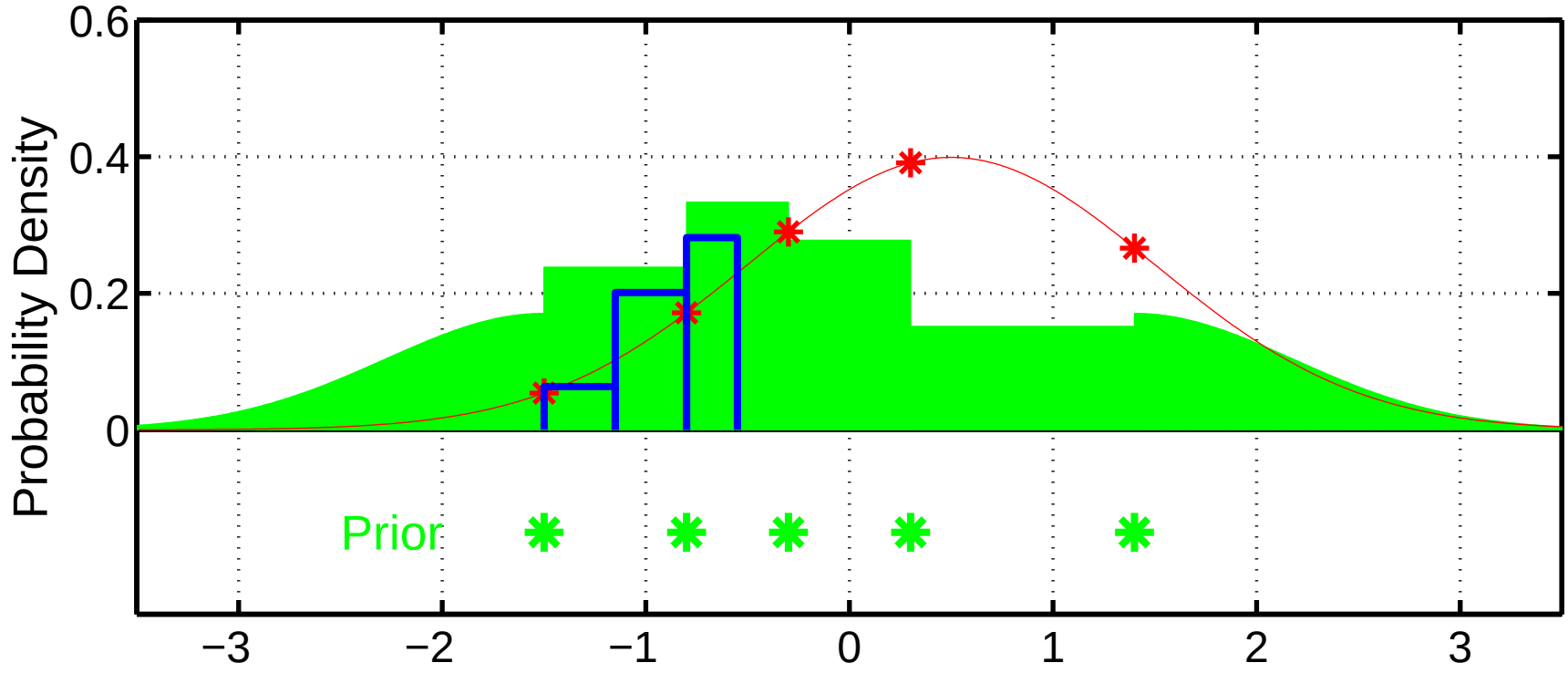


Compute posterior continuous distribution:

1. Split each uniform box in half; multiply mass by adjacent likelihood.
   (Note, outermost ensemble members only have one associated half box).

Ensemble Filter Algorithms:

Boxcar Kernel filter (filter_kind=7 in assim_tools_nml).

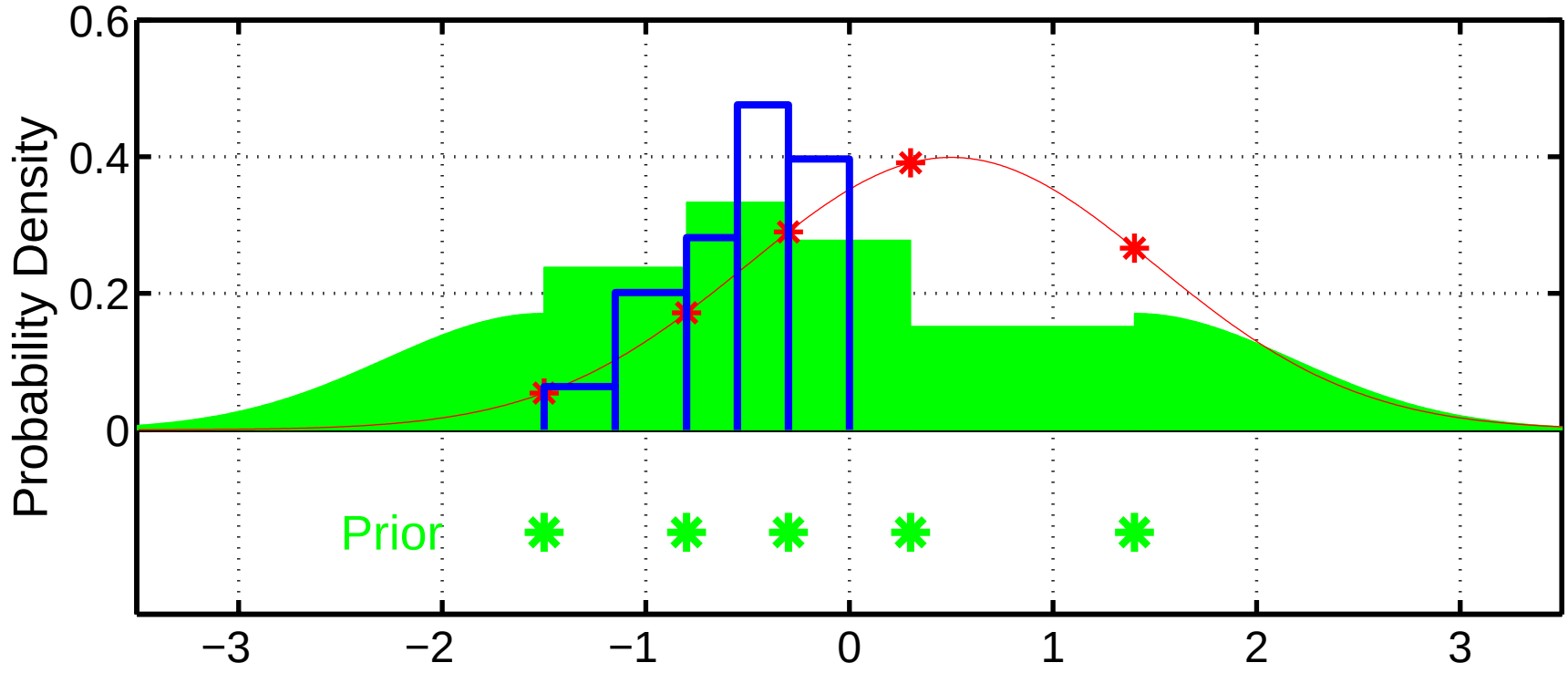


Compute posterior continuous distribution:

1. Split each uniform box in half; multiply mass by adjacent likelihood.

Ensemble Filter Algorithms:

Boxcar Kernel filter (filter_kind=7 in assim_tools_nml).

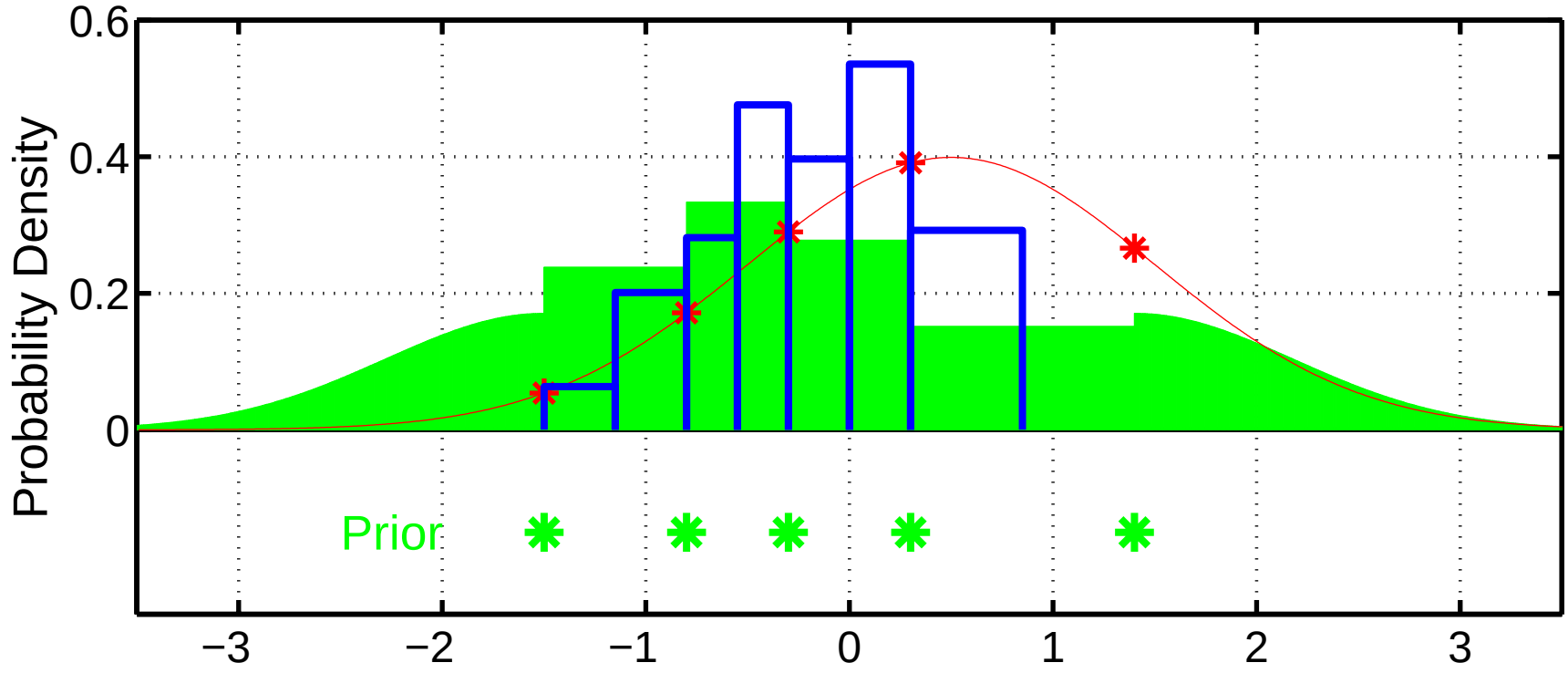


Compute posterior continuous distribution:

1. Split each uniform box in half; multiply mass by adjacent likelihood.

Ensemble Filter Algorithms:

Boxcar Kernel filter (filter_kind=7 in assim_tools_nml).

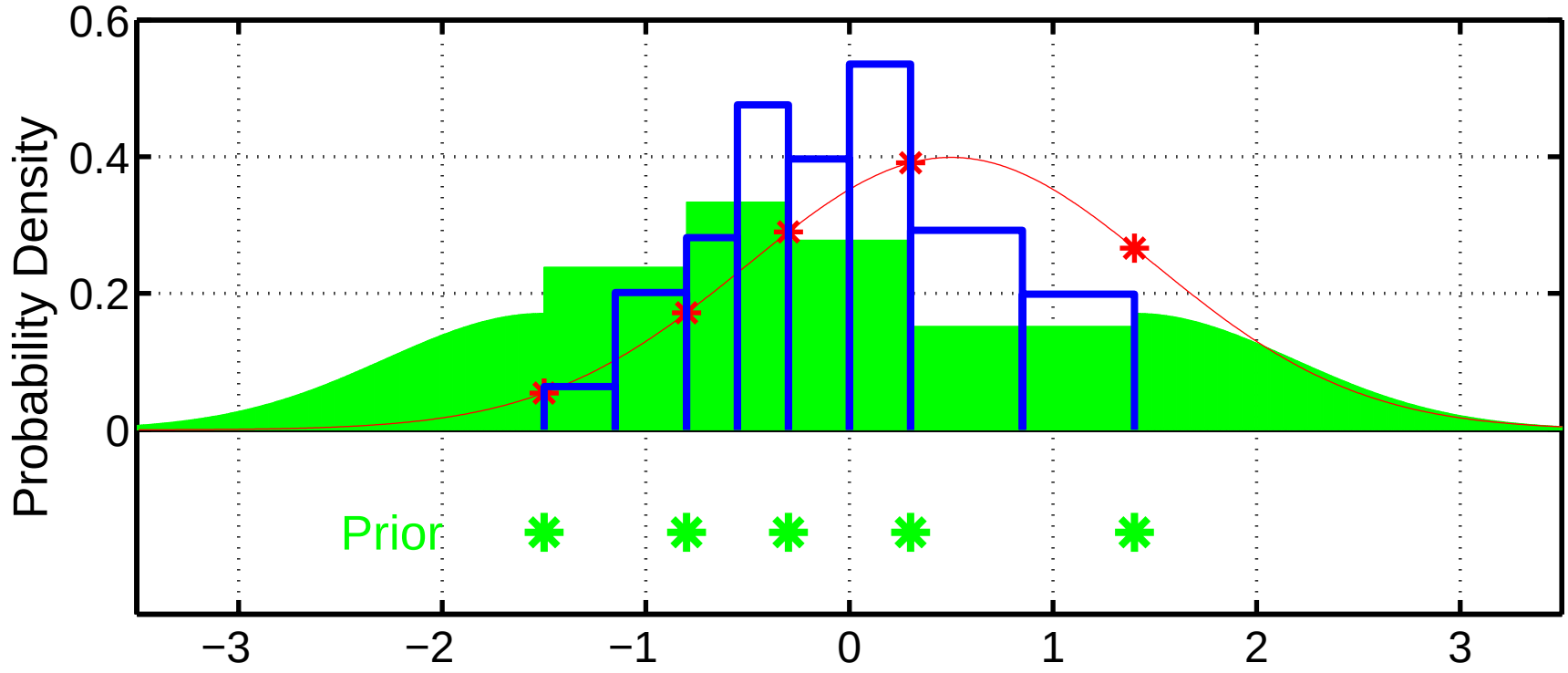


Compute posterior continuous distribution:

1. Split each uniform box in half; multiply mass by adjacent likelihood.

Ensemble Filter Algorithms:

Boxcar Kernel filter (filter_kind=7 in assim_tools_nml).

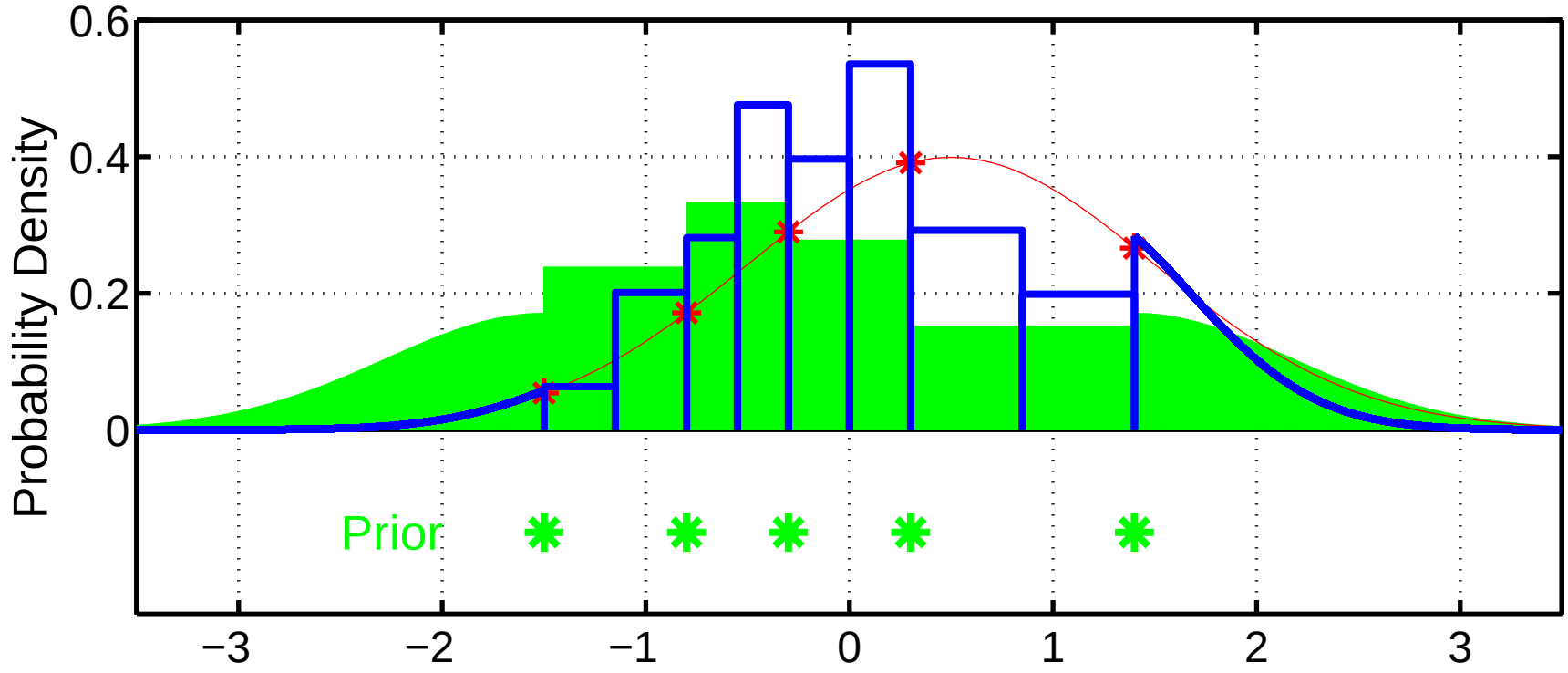


Compute posterior continuous distribution:

1. Split each uniform box in half; multiply mass by adjacent likelihood.
   (Note, outermost ensemble members only have one associated half box).

Ensemble Filter Algorithms:

Boxcar Kernel filter (filter_kind=7 in assim_tools_nml).

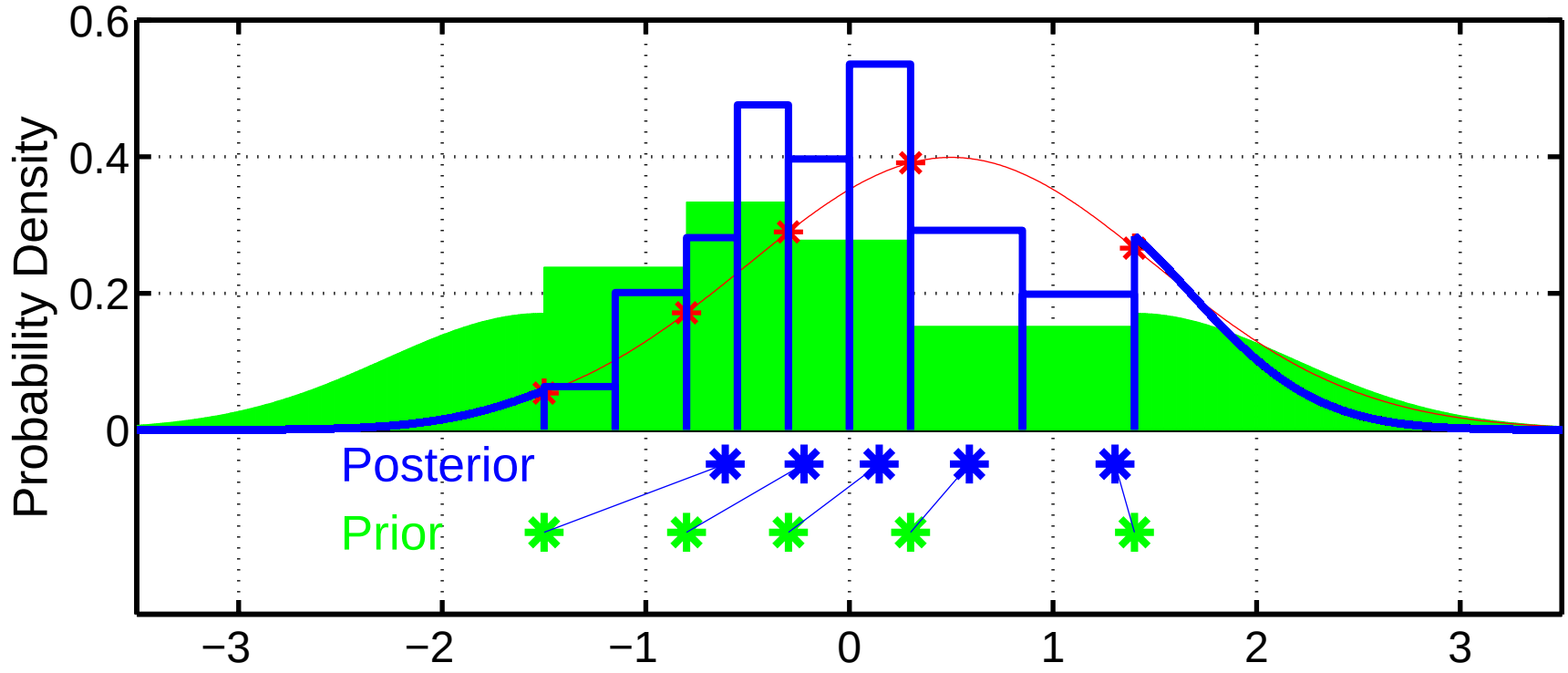


Compute posterior continuous distribution:

1. Split each uniform box in half; multiply mass by adjacent likelihood.
2. Product of prior gaussian kernel with likelihood for wings.
   Easy for gaussian likelihood; quadrature if non-gaussian.

Ensemble Filter Algorithms:

Boxcar Kernel filter (filter_kind=7 in assim_tools_nml).



Compute updated ensemble members:

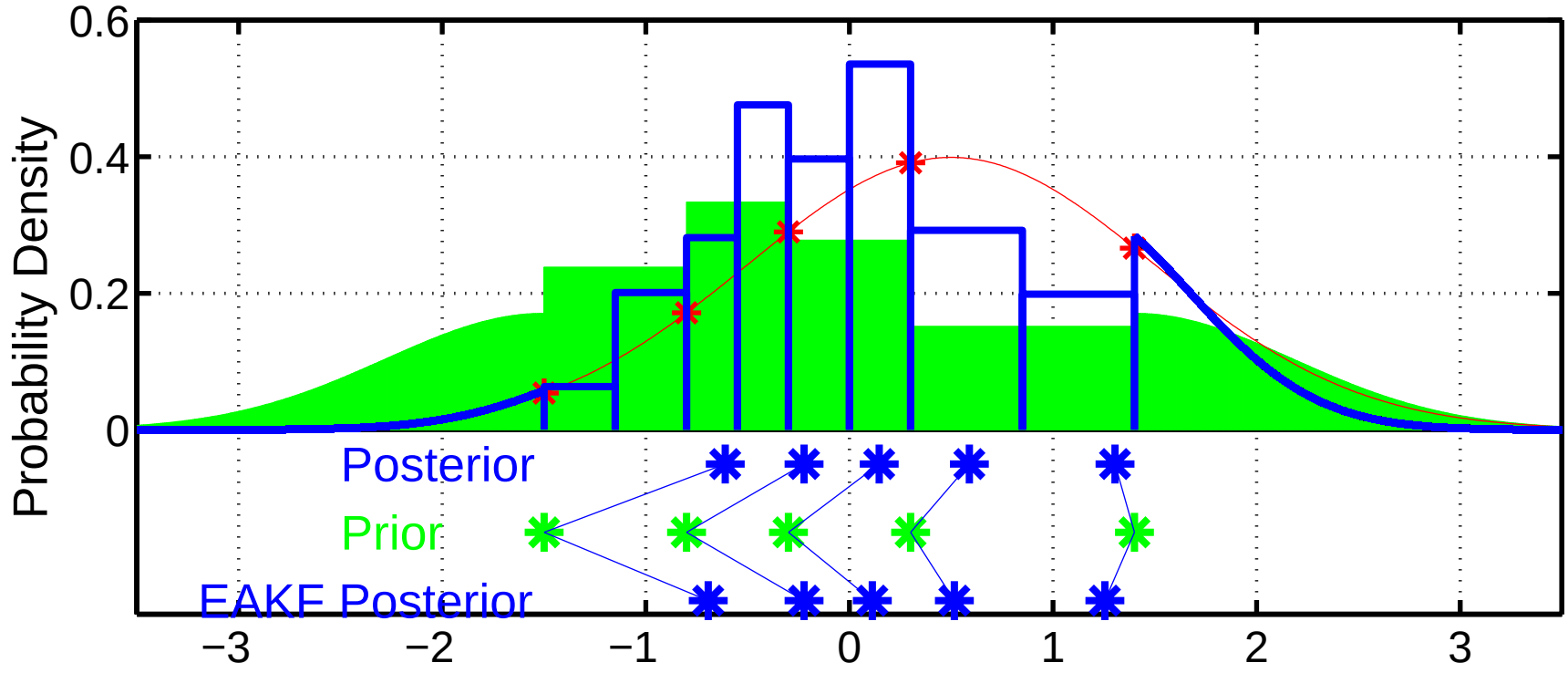
1 / (ens_size + 1) of posterior mass between each ensemble pair.

1 / (ens_size + 1) in each wing.

Trivial to compute cumulative density for if likelihood is gaussian.

Ensemble Filter Algorithms:

Boxcar Kernel filter (filter_kind=7 in assim_tools_nml).



Compare to standard (filter_kind=1) ensemble adjustment.

In this nearly gaussian case, differences in increments are small.

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 methods (*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!

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.