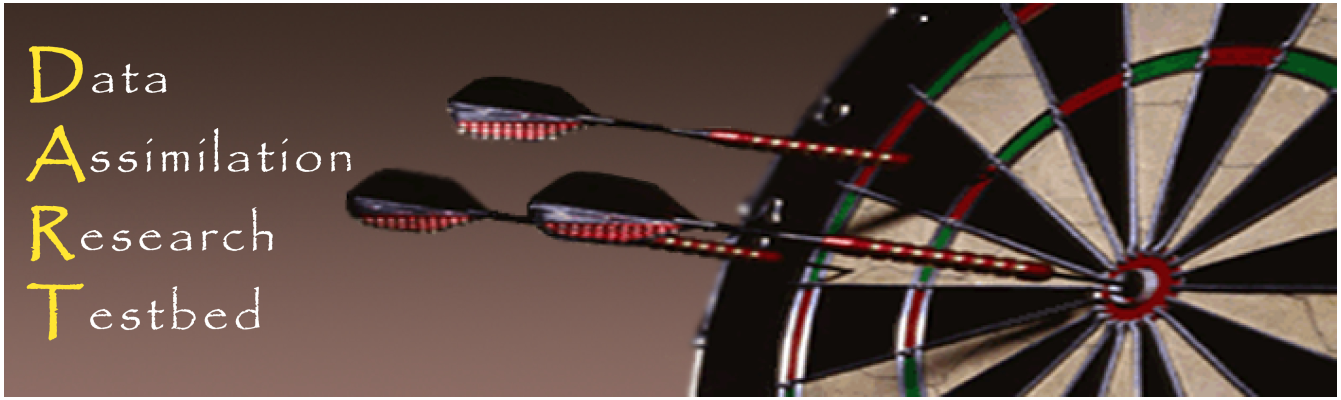
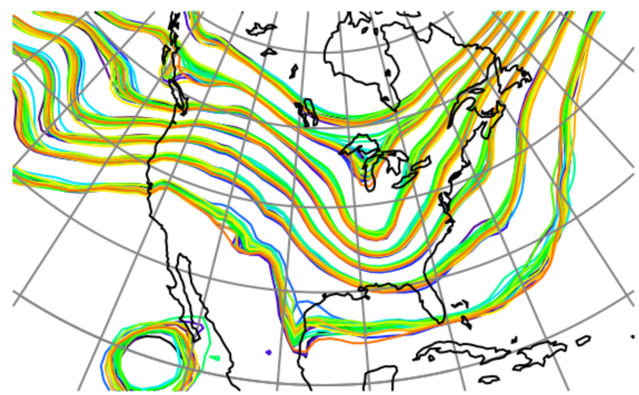


DART Tutorial Part IV:
Other Updates for an Observed Variable



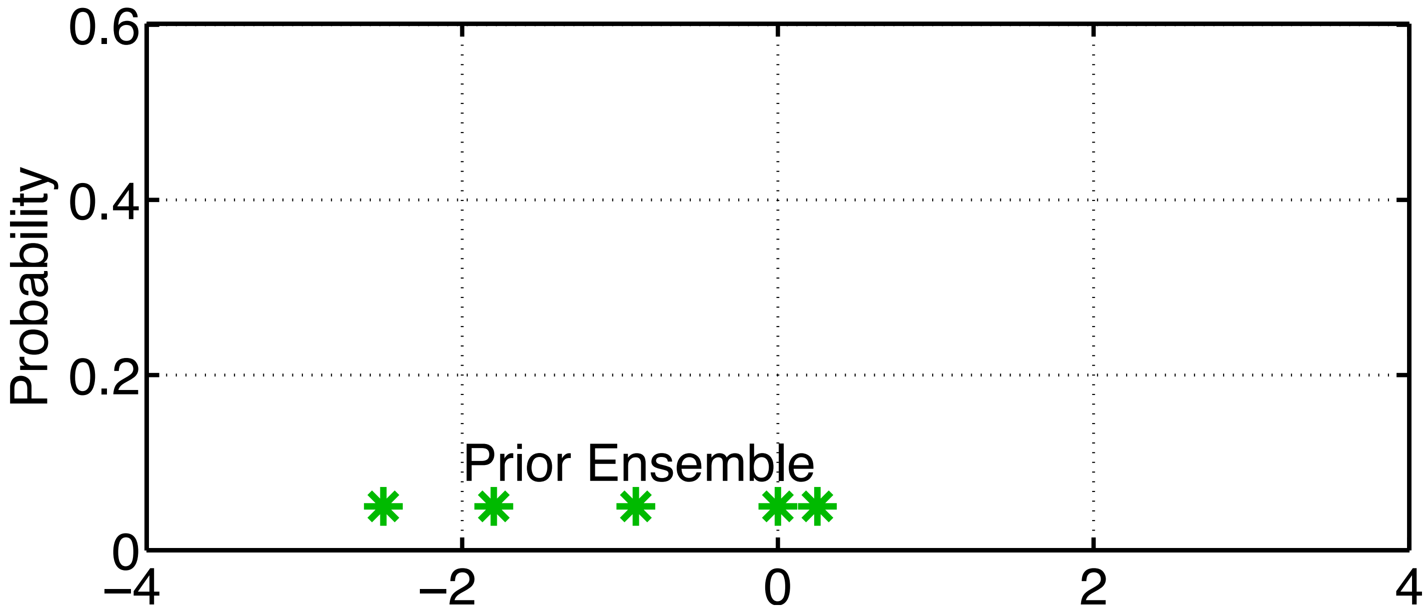
The National Center for Atmospheric Research is sponsored by the National Science Foundation.
Any opinions, findings and conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

NCAR | UCAR | National Center for Atmospheric Research

Product of Two Gaussians

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

Ensemble filters: Prior is available as finite sample.

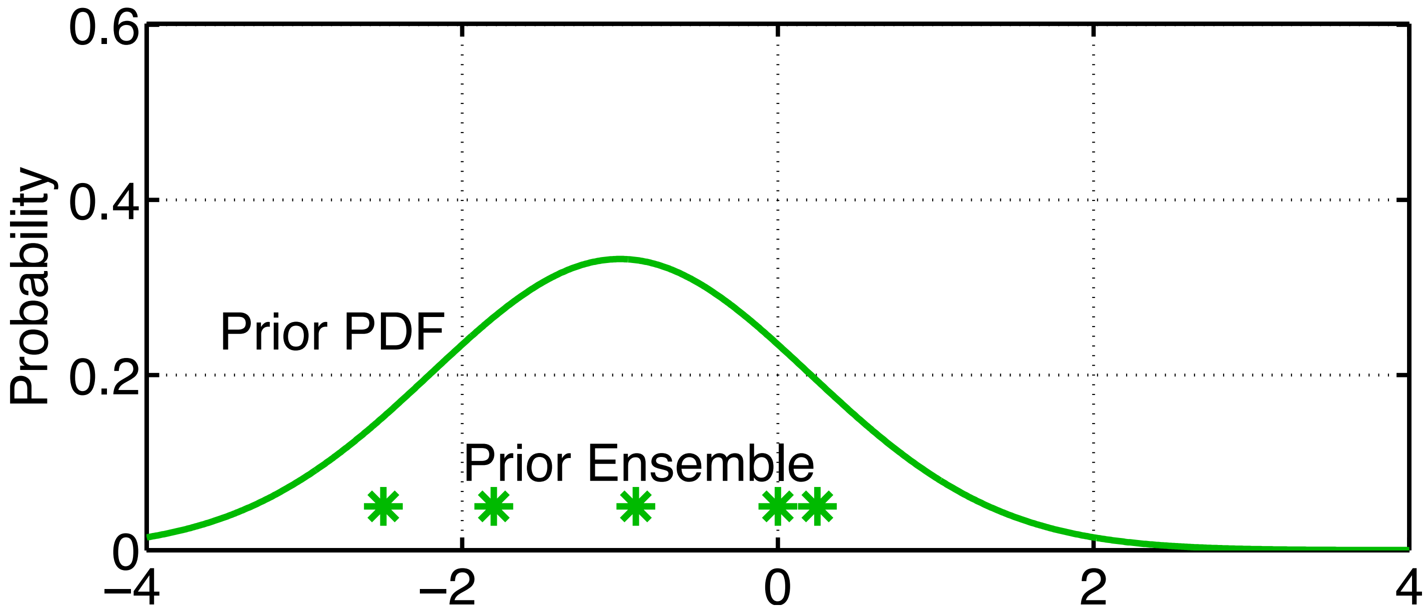


Don’t know much about properties of this sample.
May naively assume it is random draw from ‘truth’.

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?

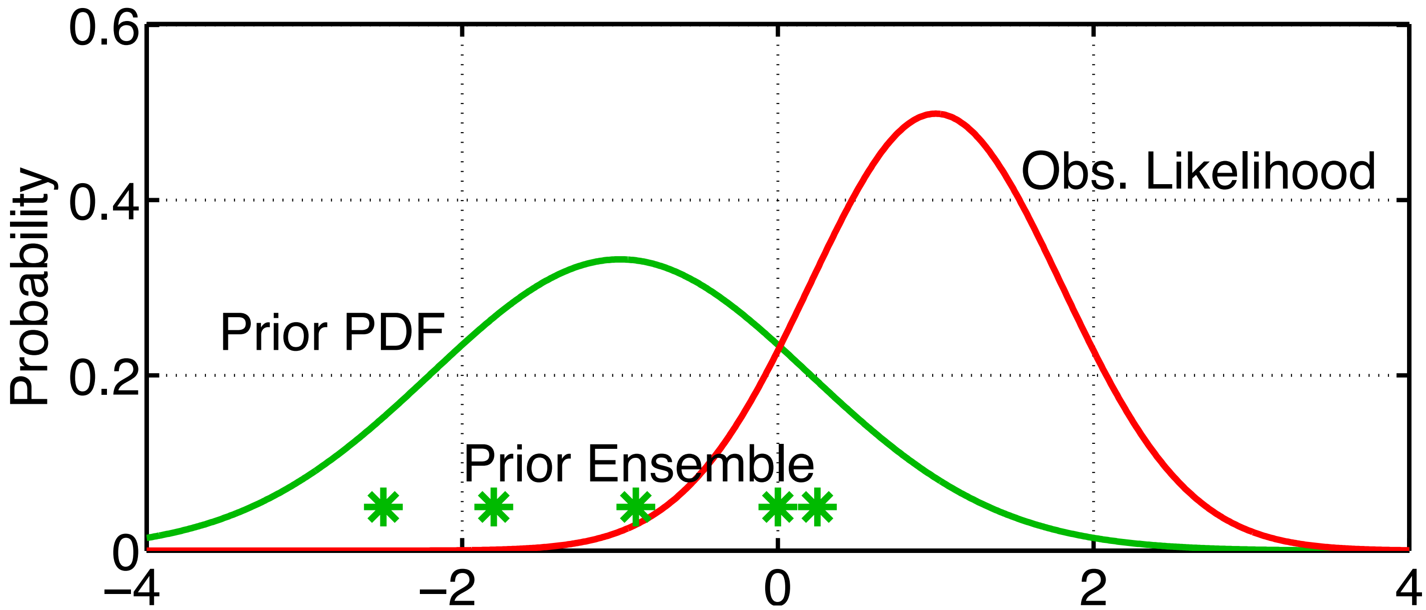


Fit a continuous (Gaussian for now) distribution to sample.

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).

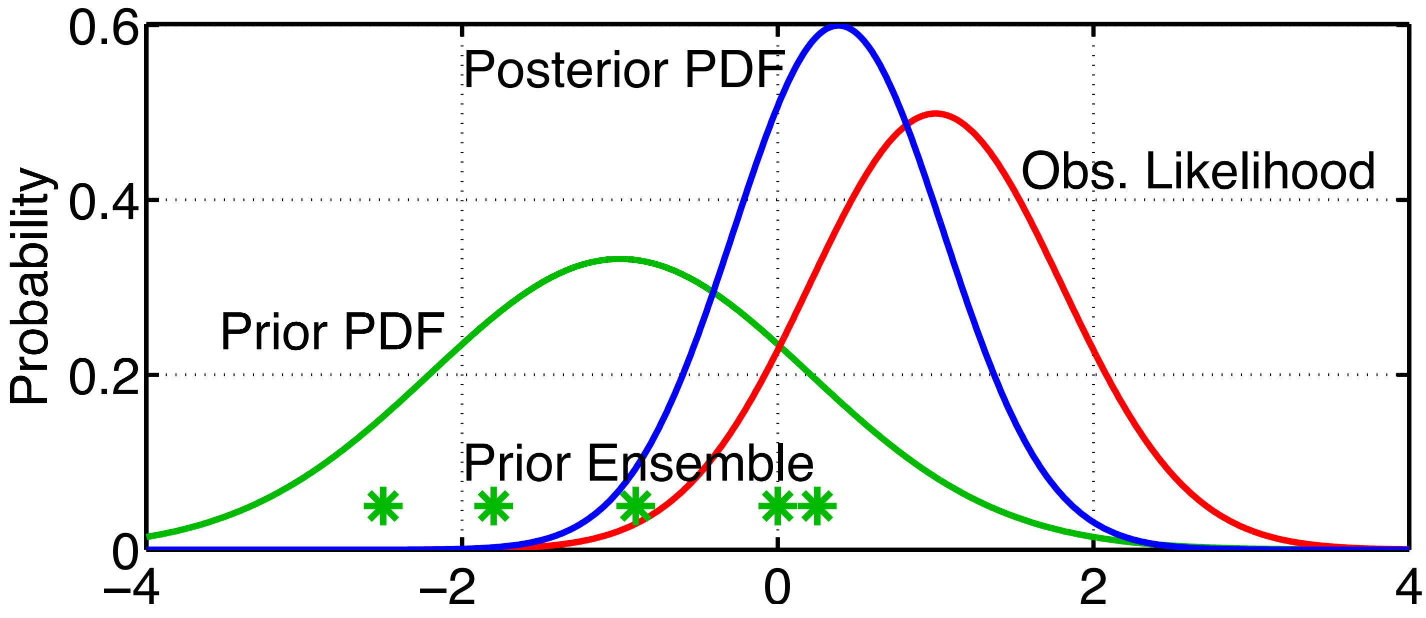


If Obs. likelihood isn't Gaussian, can generalize methods below.

Product of Two Gaussians

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

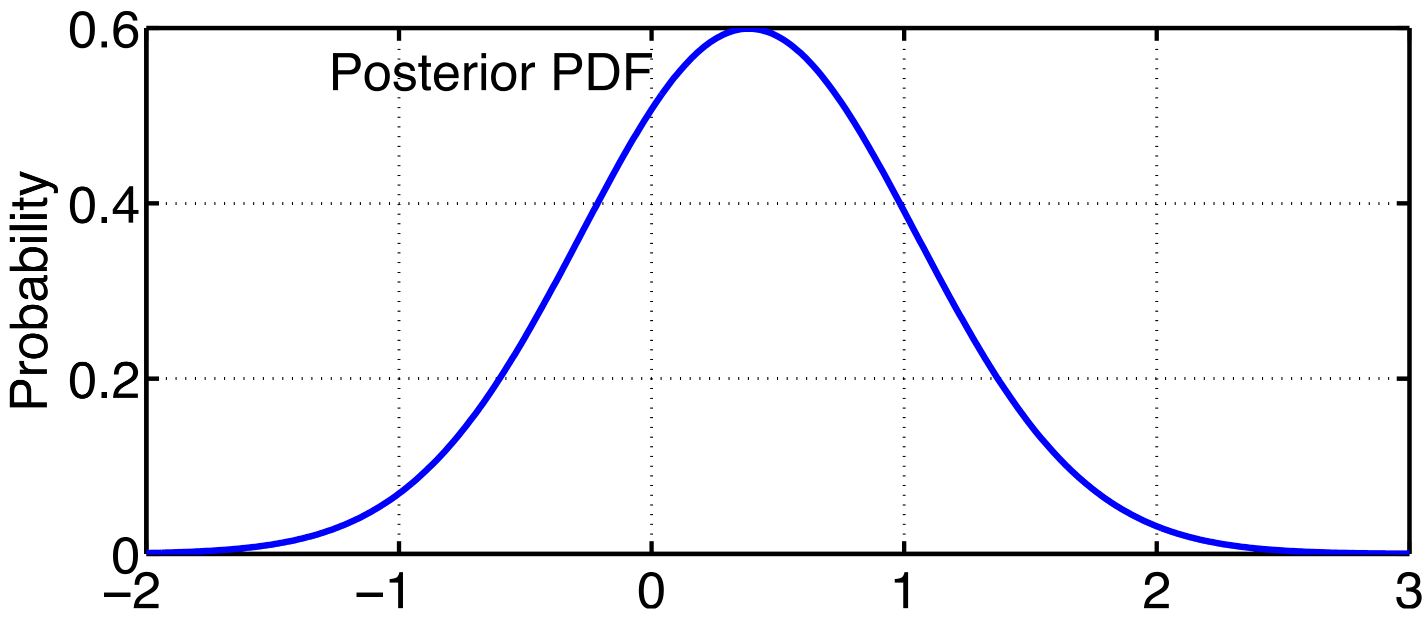
Product of prior Gaussian fit and Obs. likelihood is Gaussian.



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

Sampling Posterior PDF:

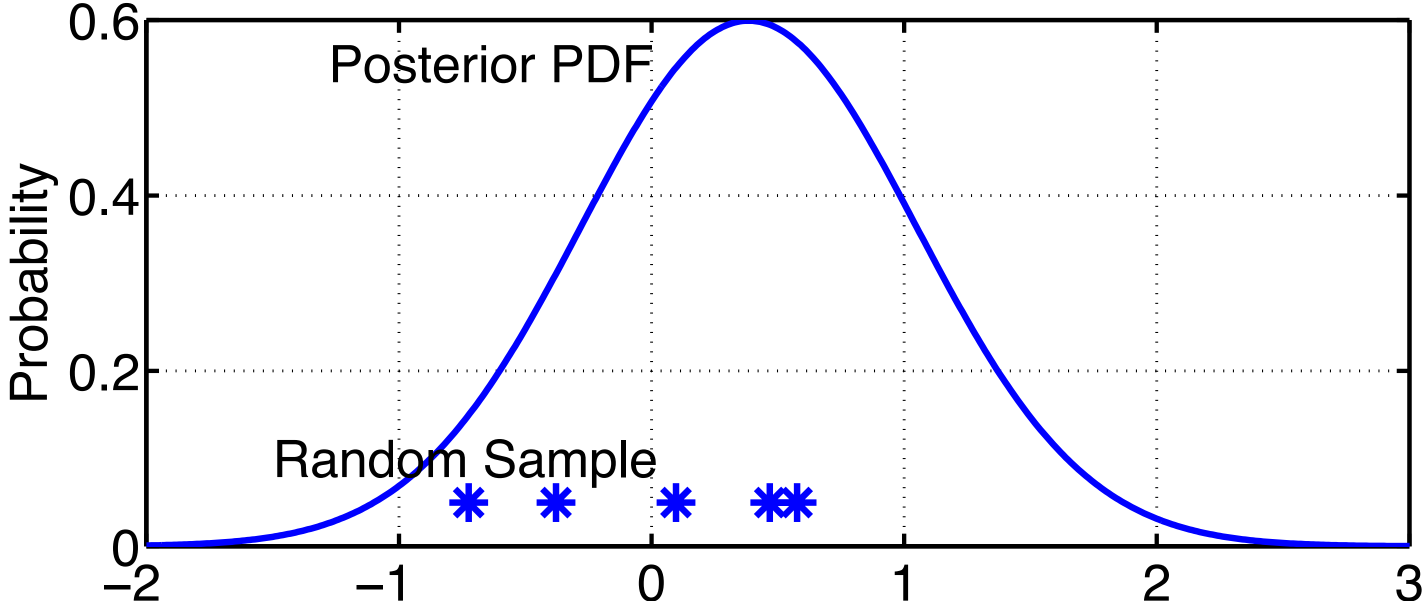
There are many ways to do this.



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.

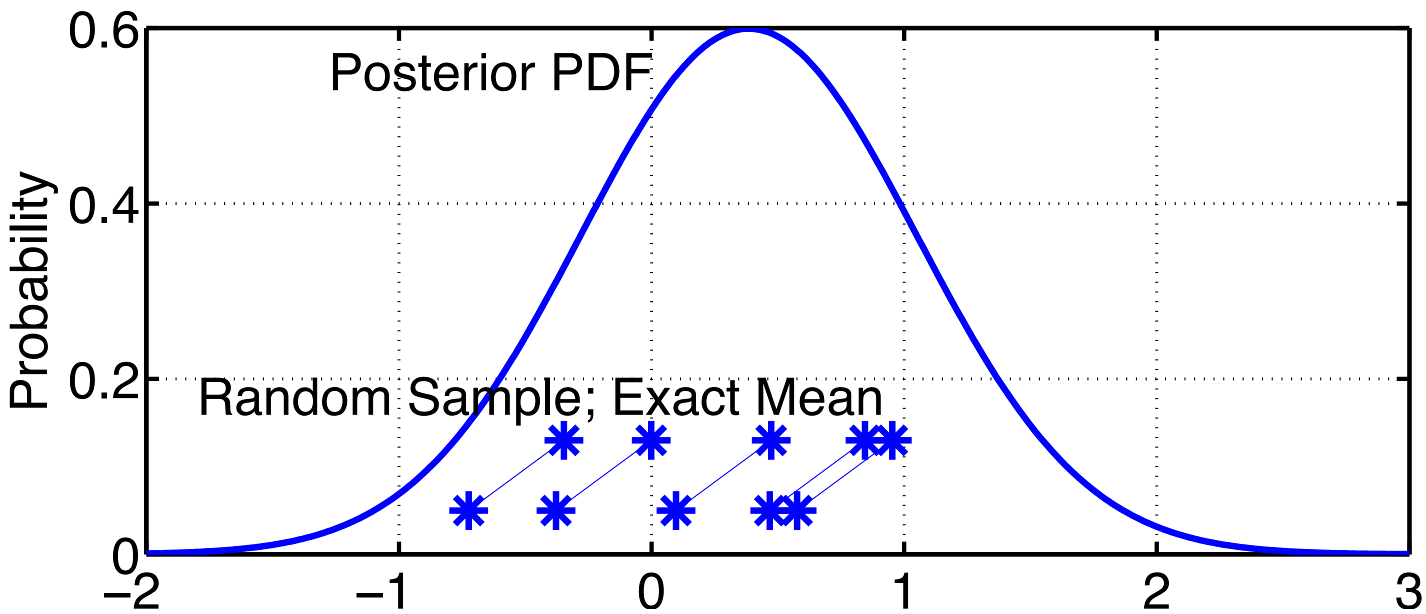
Sampling Posterior PDF:

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



Sampling Posterior PDF:

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

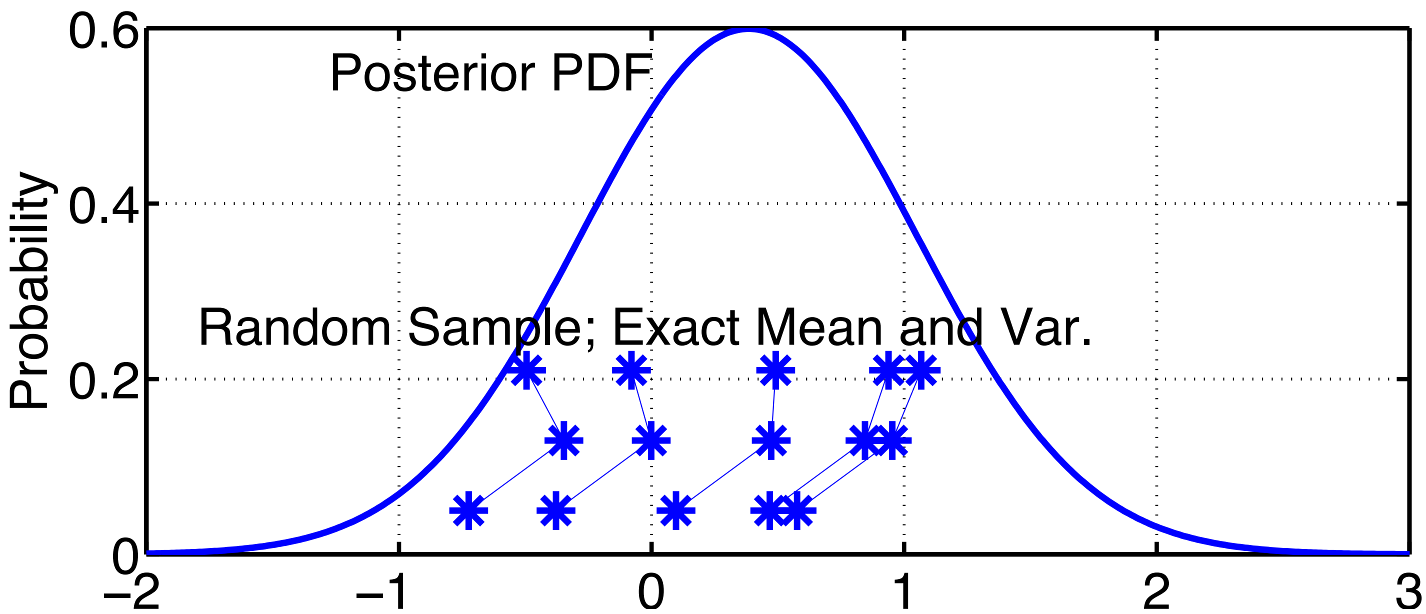


Can ‘play games’ with this sample to improve (modify) its properties.

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`).

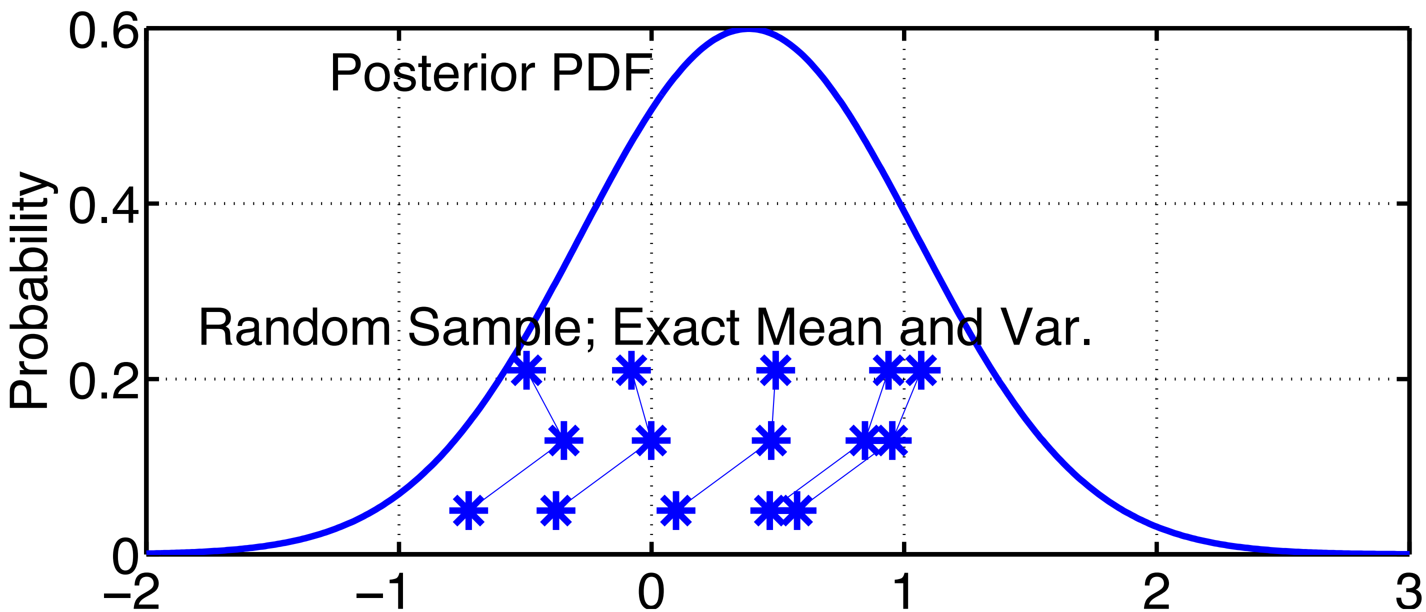


Can 'play games' with this sample to improve (modify) its properties.

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 (*filter_kind=5* in `&assim_tools_nml`).

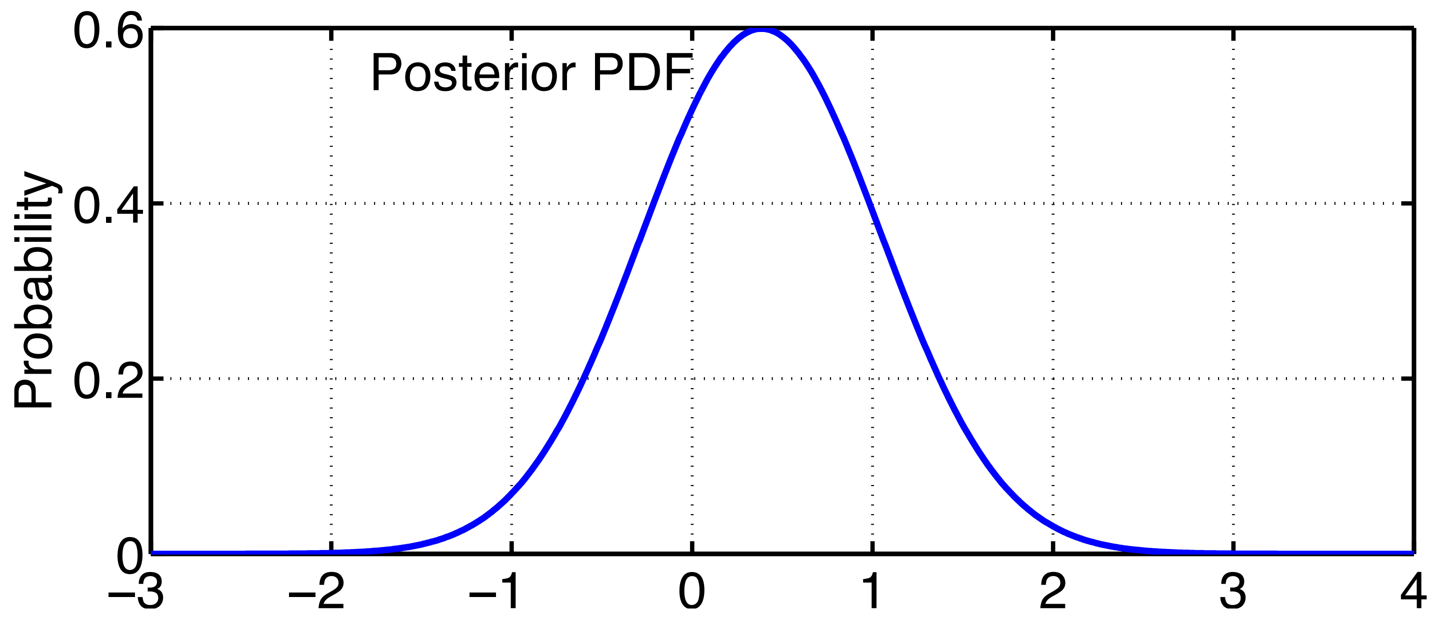


Might also want to eliminate rare extreme outliers.

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.

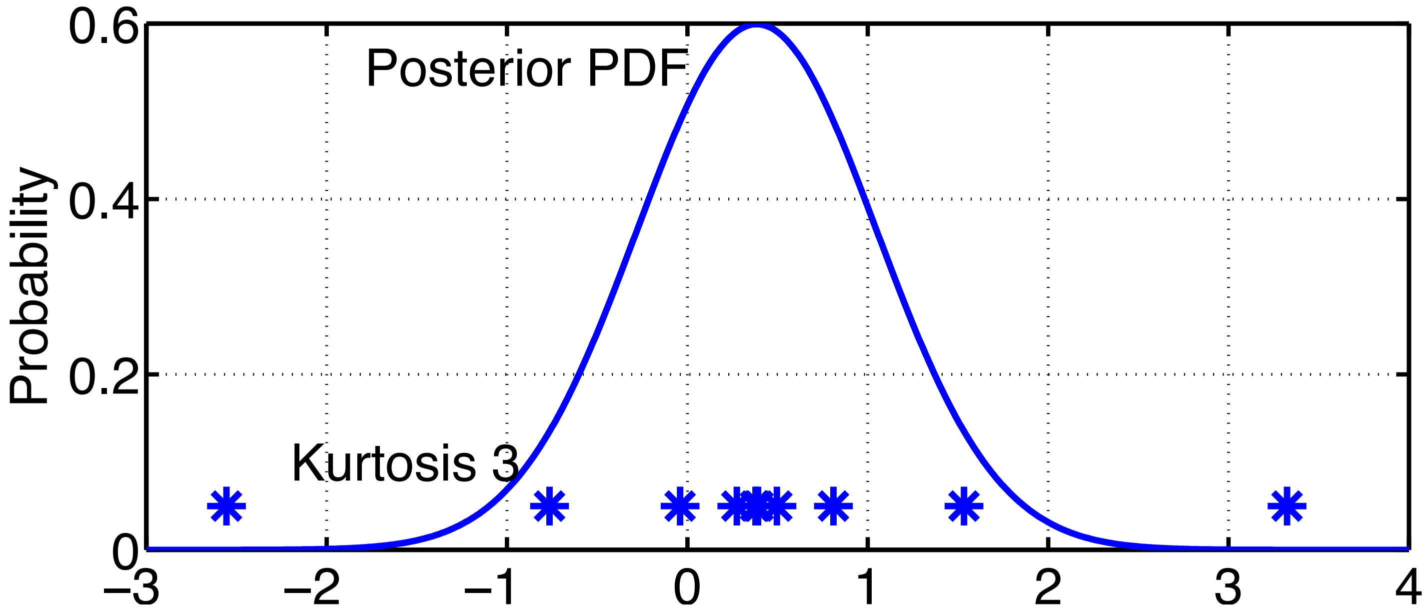


For instance: Sample could have exact mean and variance.

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).

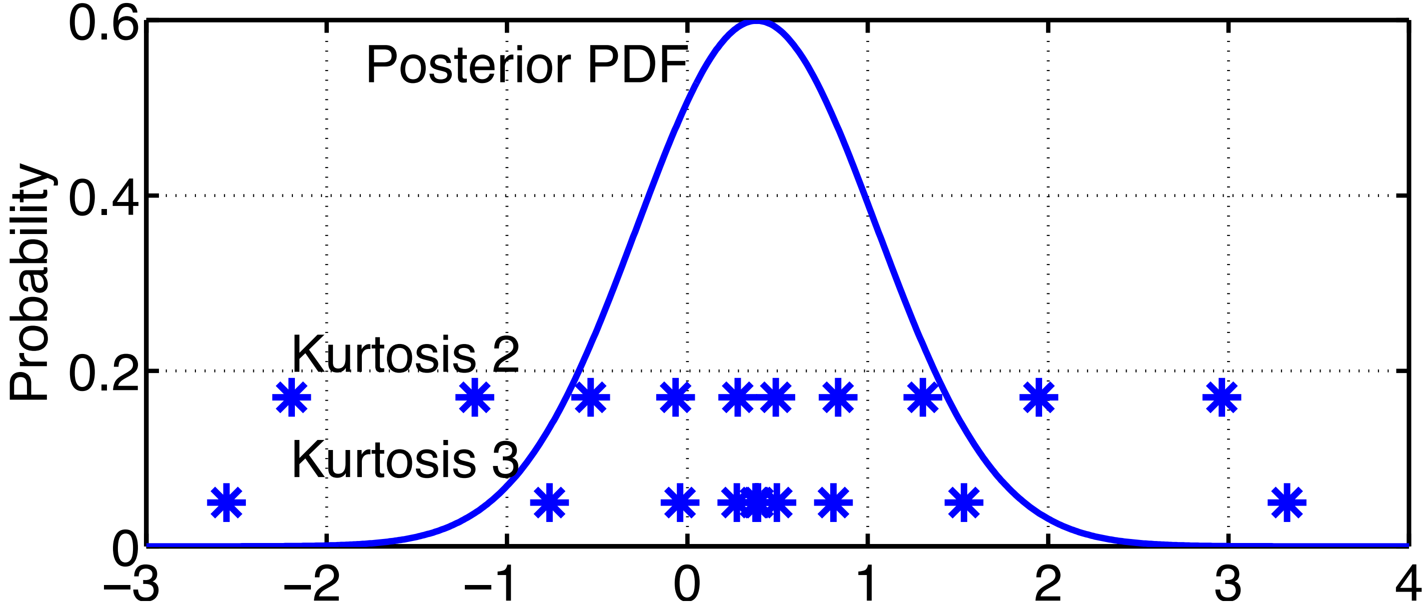


Example: Exact sample mean and variance.

Sample kurtosis (related to the sharpness/tailedness of a distribution) is 3, which is the expected value for a normal distribution. Start by assuming a uniformly-spaced sample and adjusting quadratically.

Sampling Posterior PDF:

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

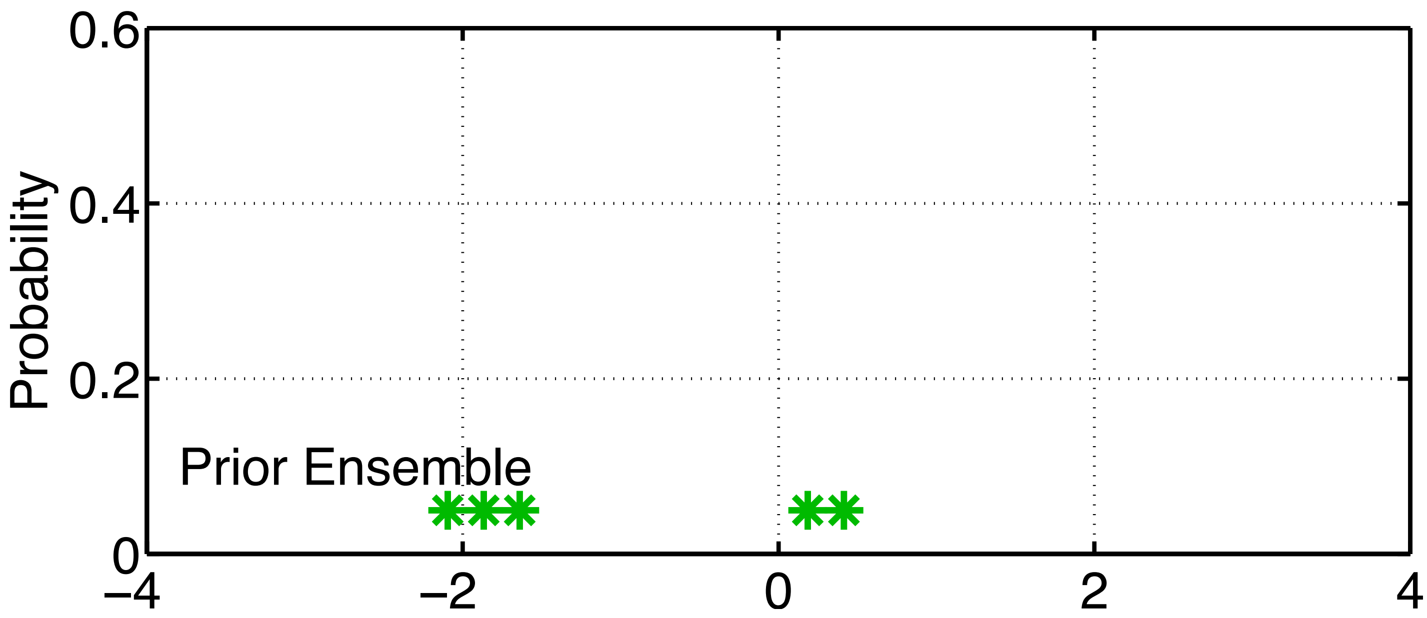


Example: Exact sample mean and variance.

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:

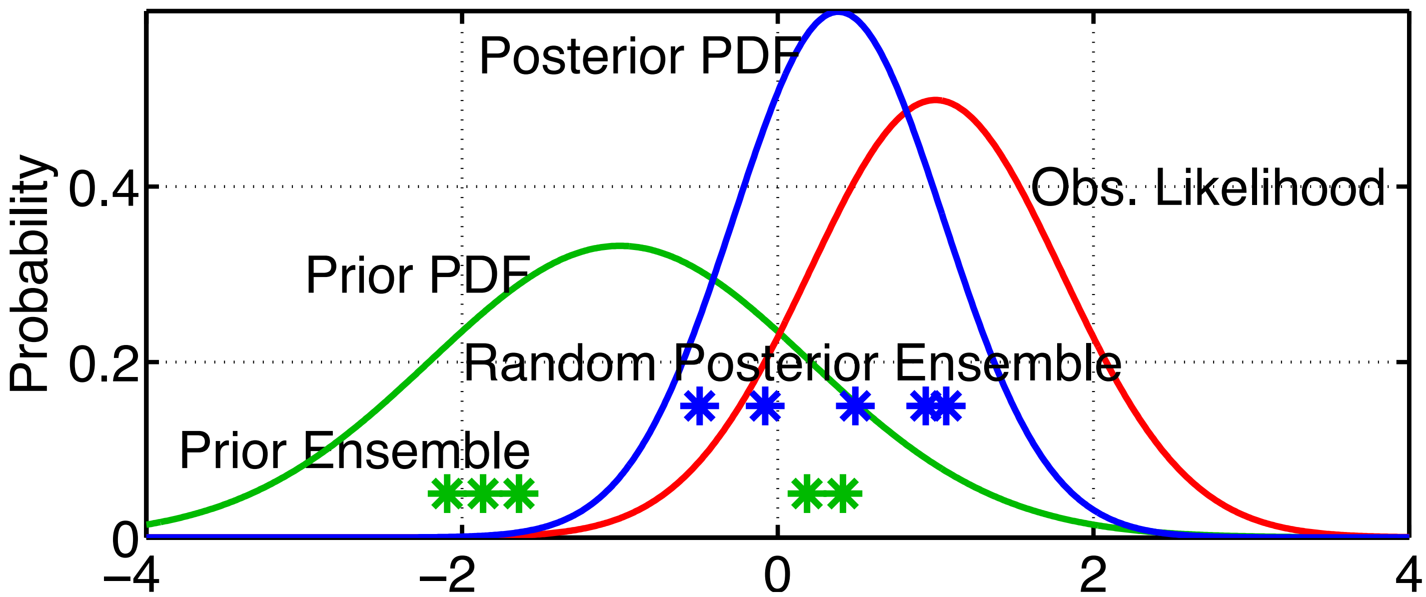
First two methods depend only on mean and variance of prior sample.



Example: Suppose prior sample is (significantly) bimodal?

Sampling Posterior PDF:

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

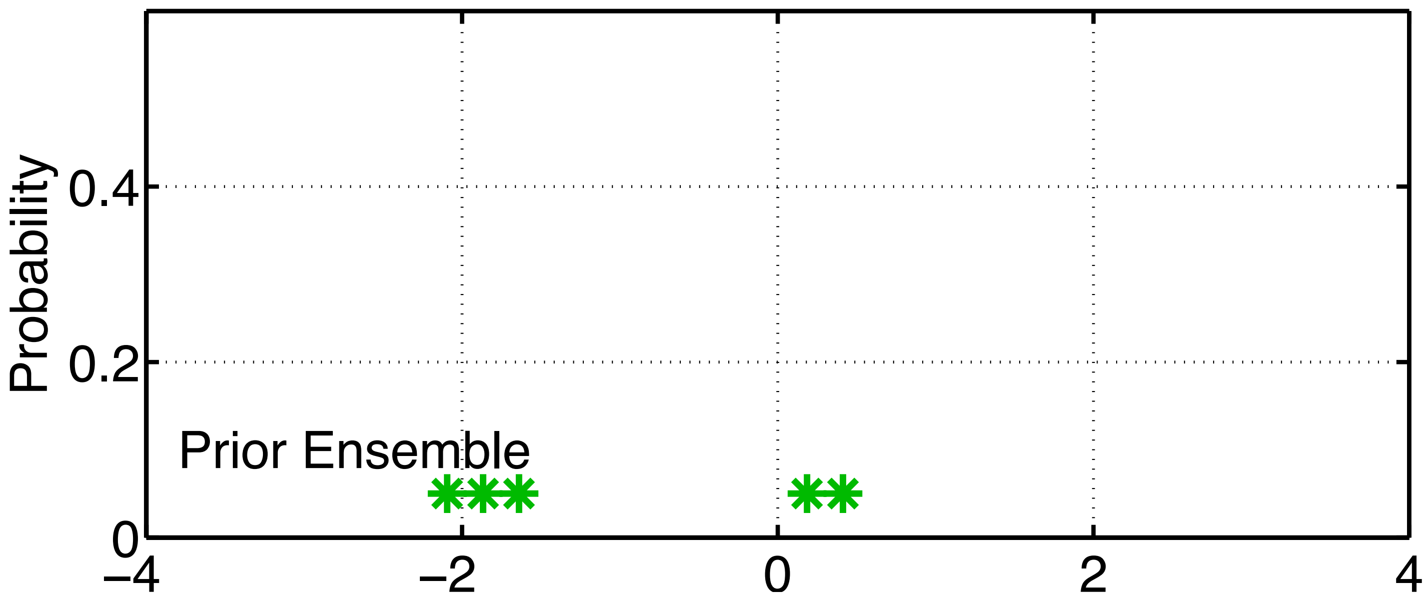


Example: Suppose prior sample is (significantly) bimodal?

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`).

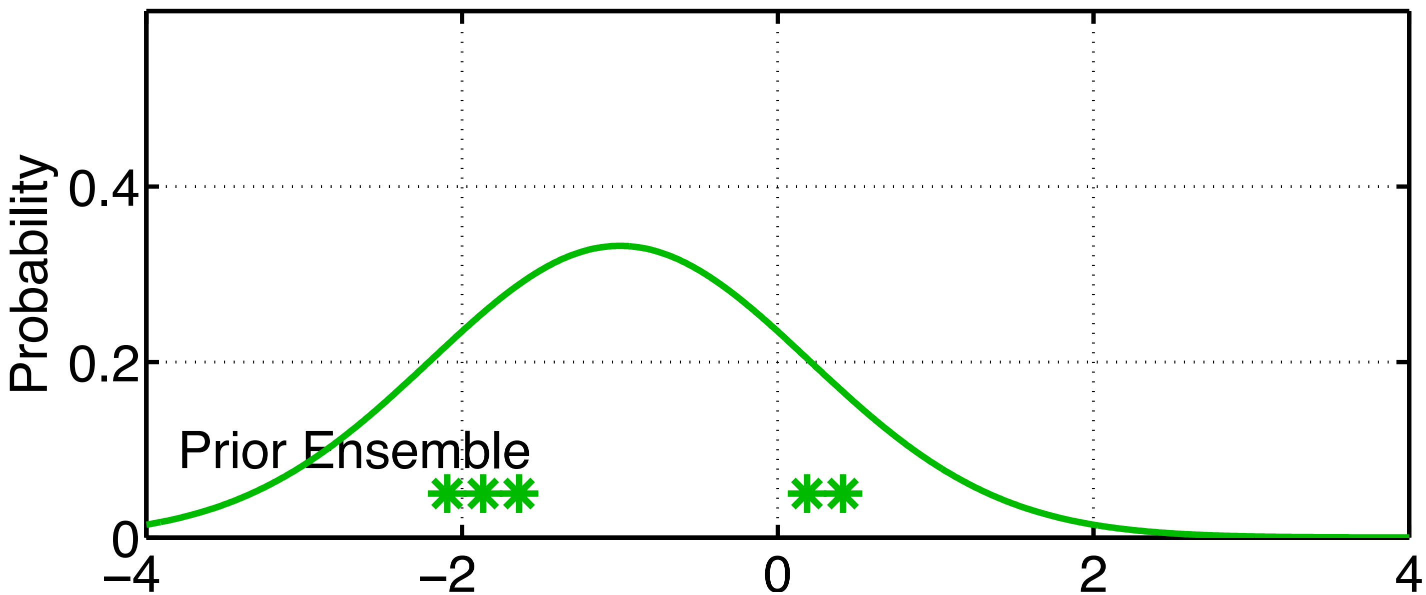


'Classical' Monte Carlo algorithm for data assimilation

Ensemble Filter Algorithms:

Ensemble Kalman Filter (EnKF)

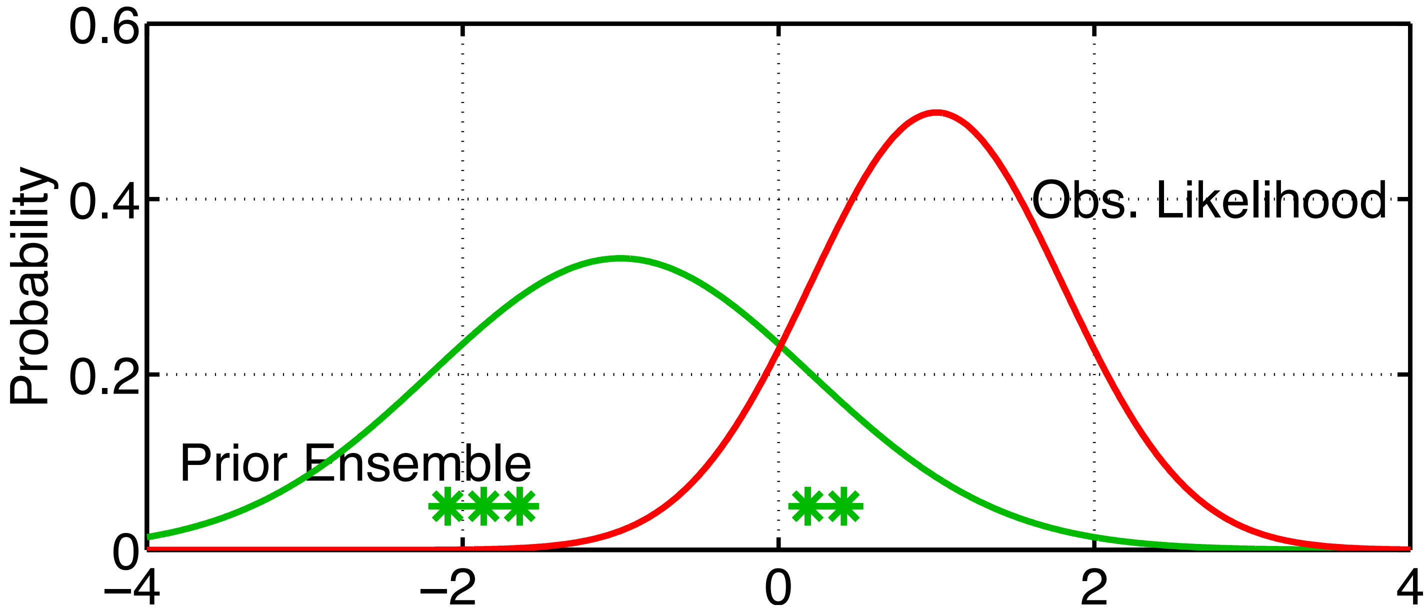
(*filter_kind*=2 in `&assim_tools_nml`).



Again, fit a Gaussian to the sample.

Ensemble Filter Algorithms:

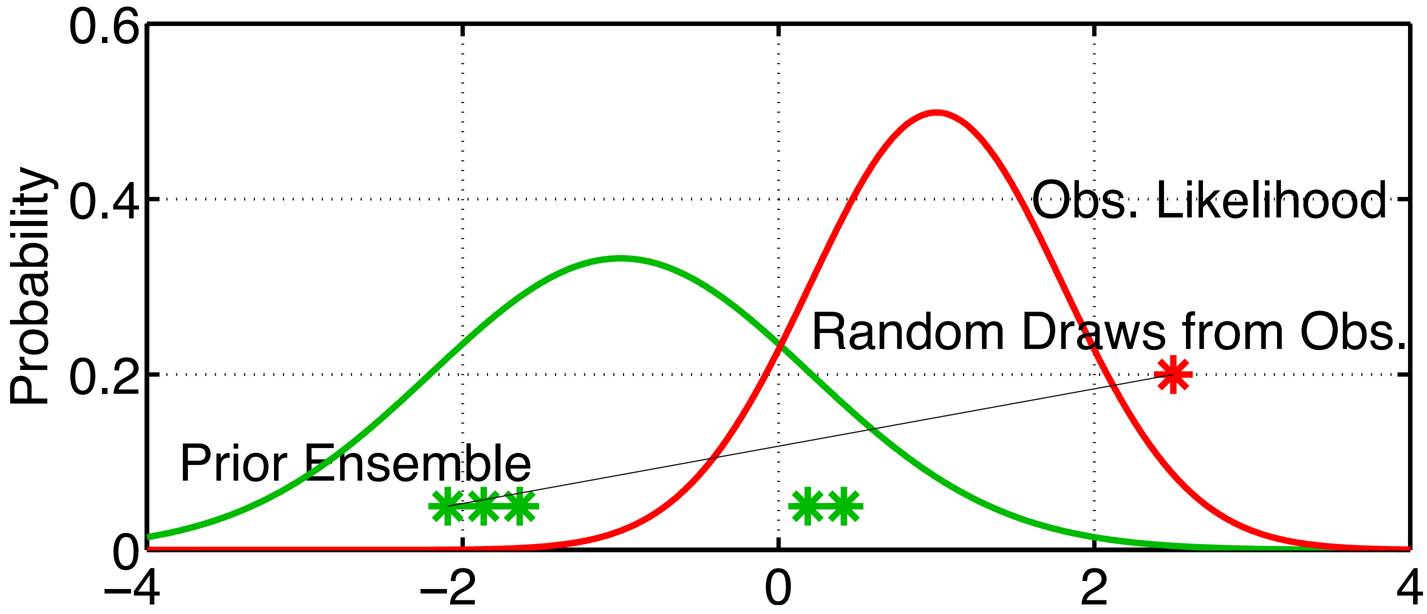
Ensemble Kalman Filter (EnKF)
(*filter_kind*=2 in `&assim_tools_nml`).



Again, fit a Gaussian to the sample.

Ensemble Filter Algorithms:

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

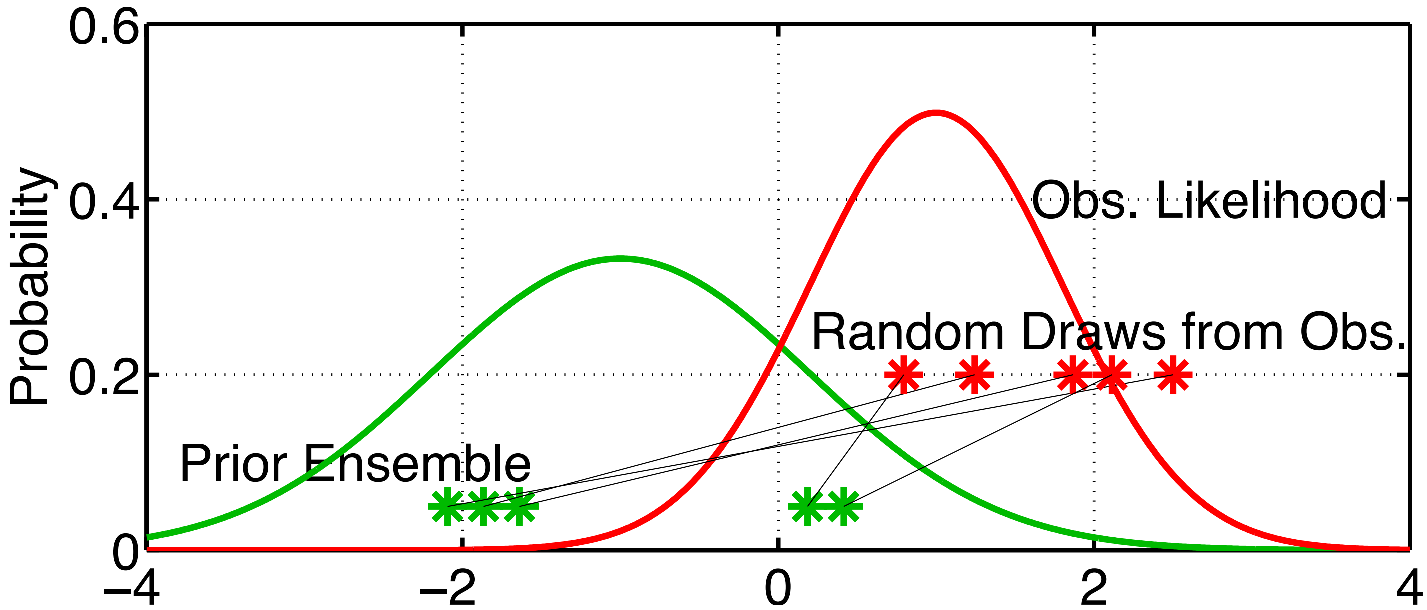


Generate a random draw from the observation likelihood.
Associate it with the first sample of the prior ensemble.

Ensemble Filter Algorithms:

Ensemble Kalman Filter (EnKF)

(*filter_kind*=2 in `&assim_tools_nml`).



Have sample of joint prior distribution for observation and prior MEAN.

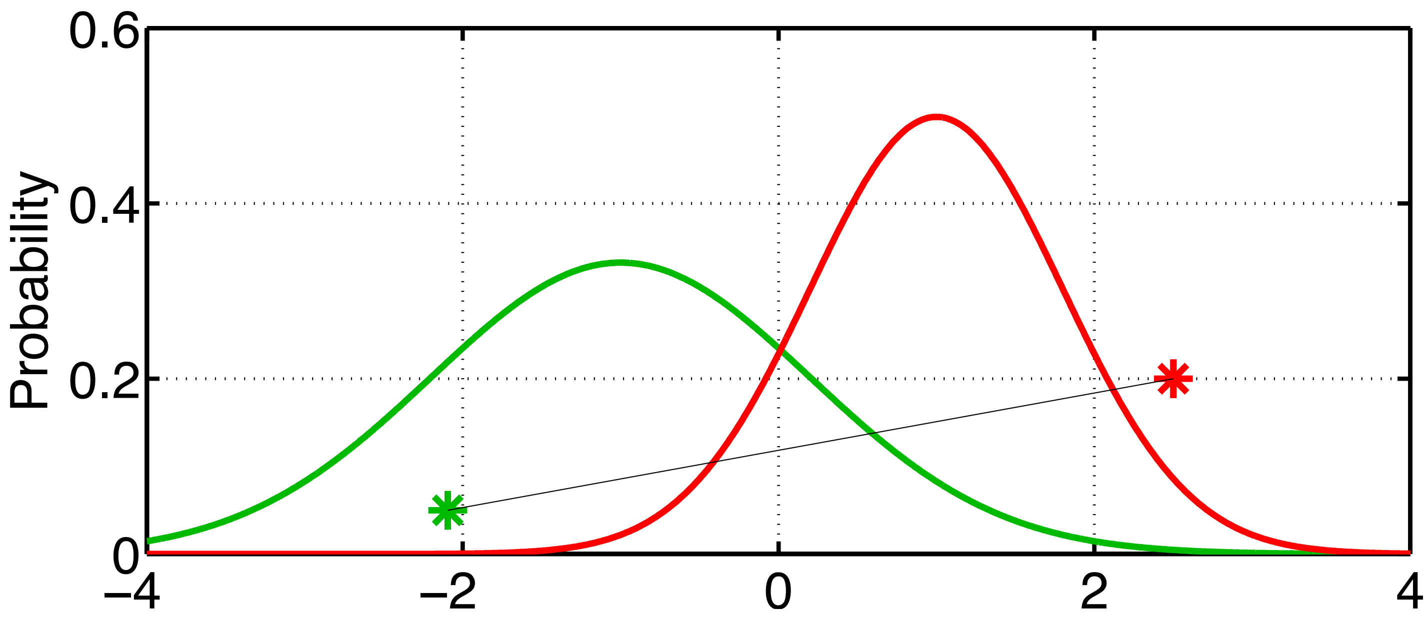
Adjusting the mean of obs. sample to be exact improves performance.

Adjusting the variance may further improve performance.

Outliers are a potential problem, but can be removed.

Ensemble Filter Algorithms:

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

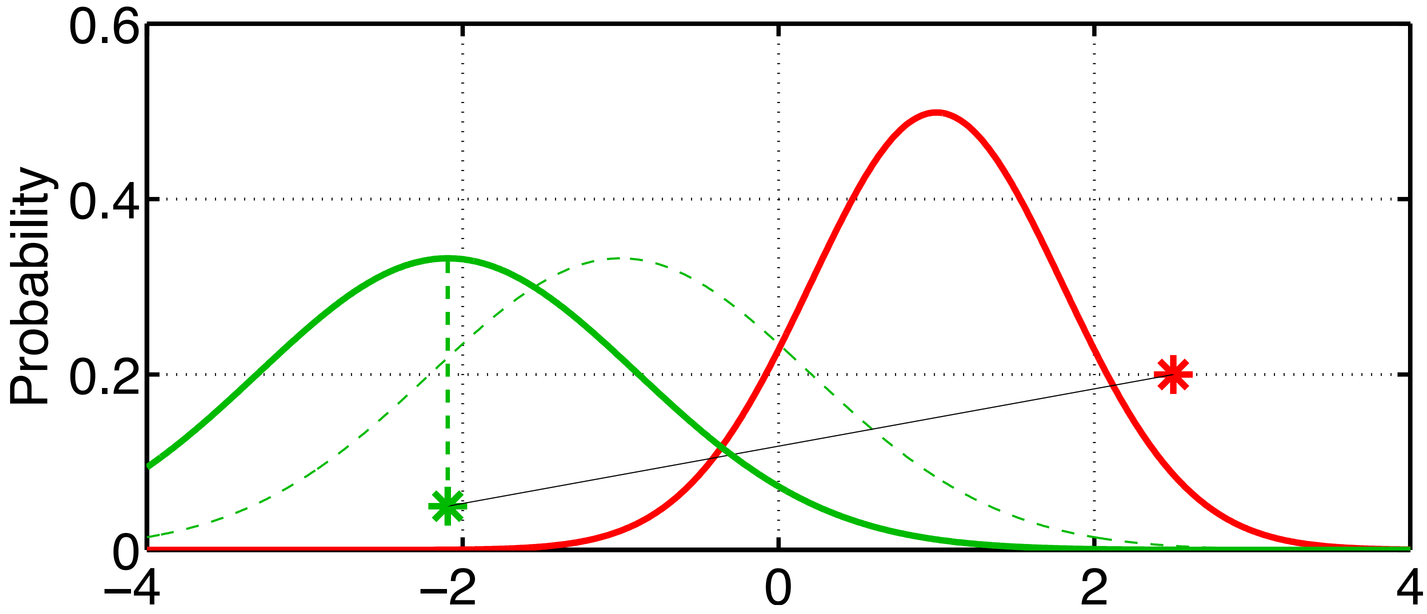


For each prior mean/obs. pair, find mean of posterior PDF.

Ensemble Filter Algorithms:

Ensemble Kalman Filter (EnKF)

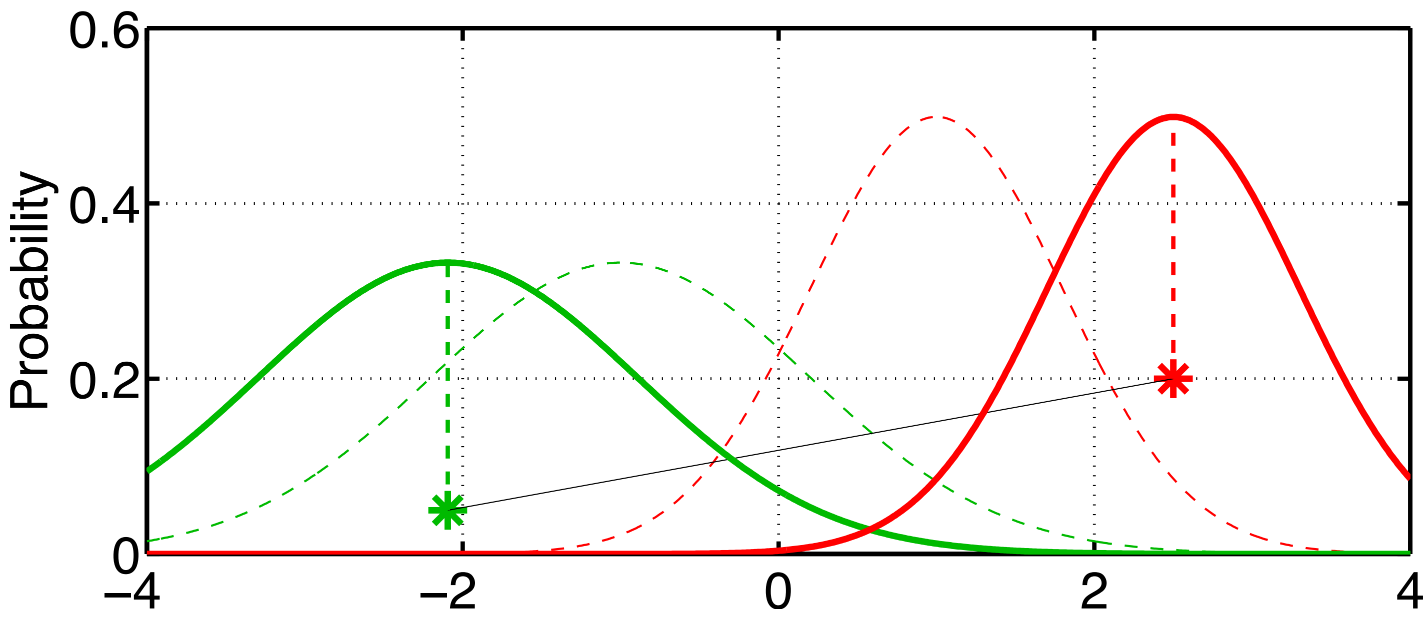
(*filter_kind*=2 in `&assim_tools_nml`).



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`).

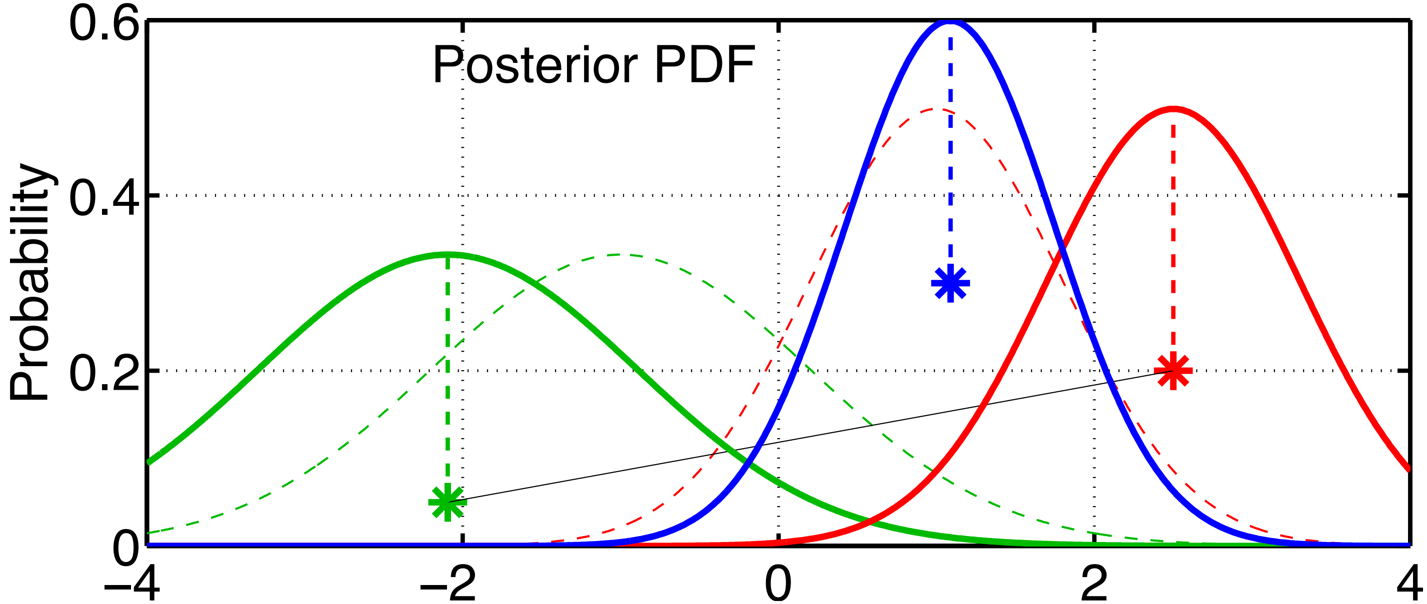


Prior sample standard deviation still measures uncertainty of prior mean estimate.

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

Ensemble Filter Algorithms:

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

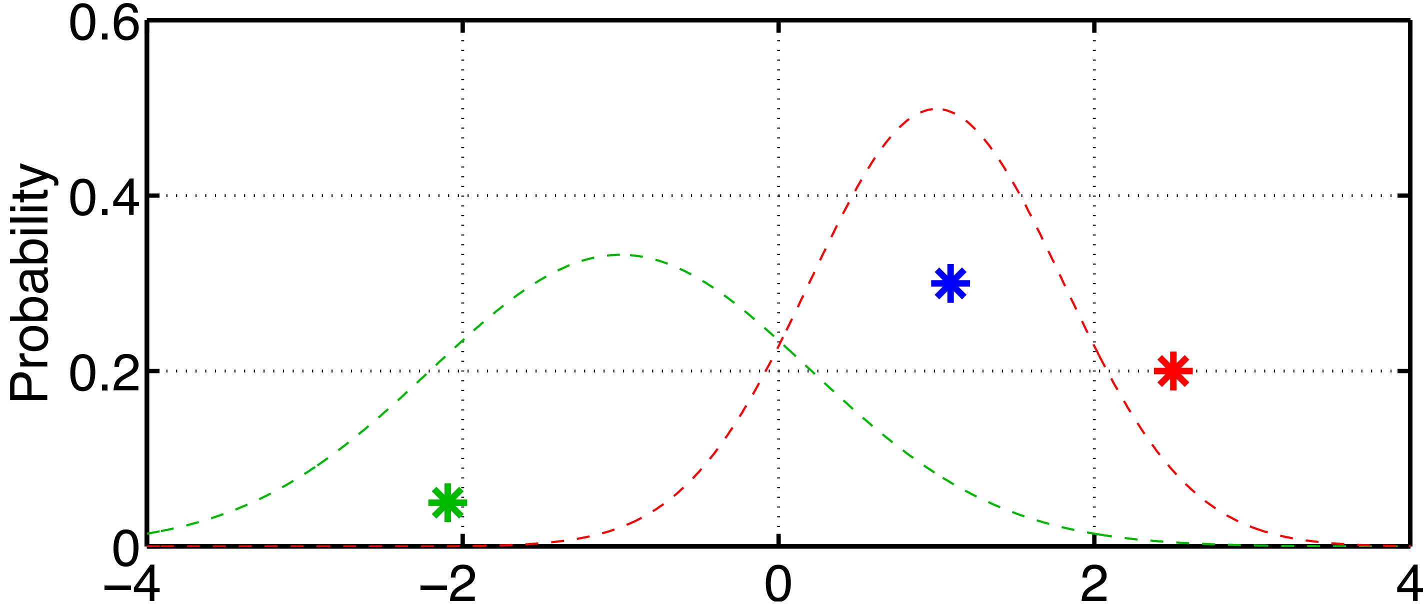


Take product of the prior/obs distributions for first sample.
This is the standard Gaussian product.

Ensemble Filter Algorithms:

Ensemble Kalman Filter (EnKF)

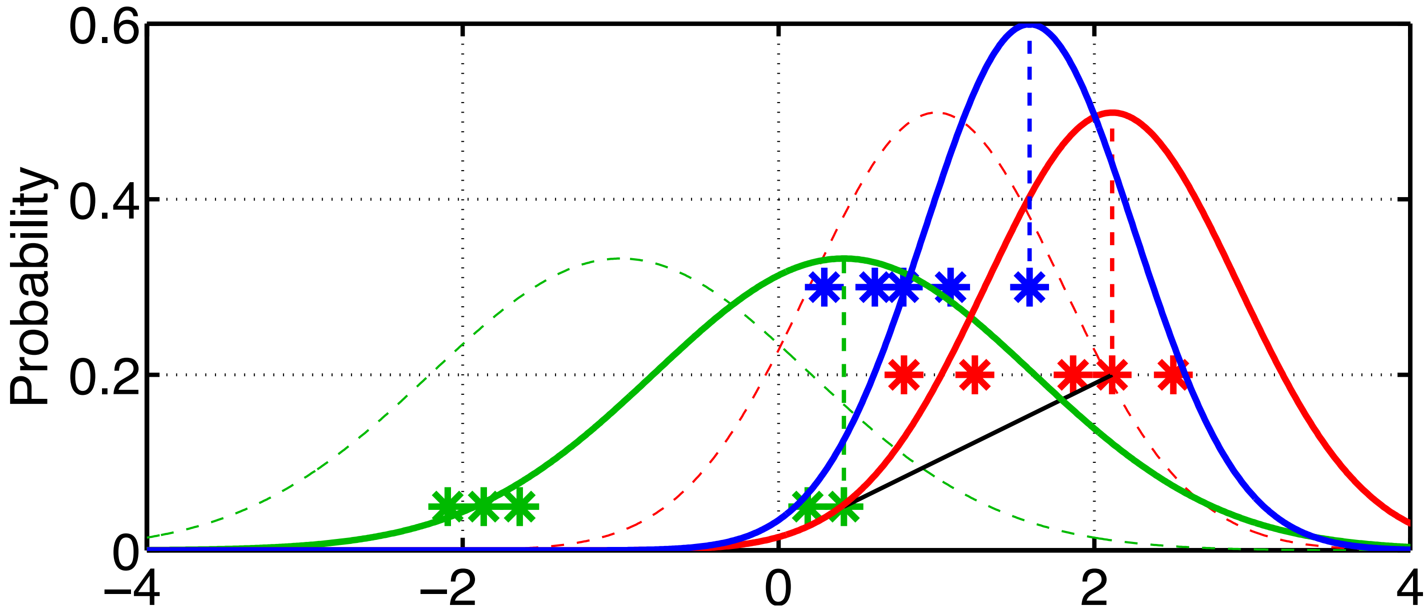
(*filter_kind*=2 in `&assim_tools_nml`).



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

Ensemble Filter Algorithms:

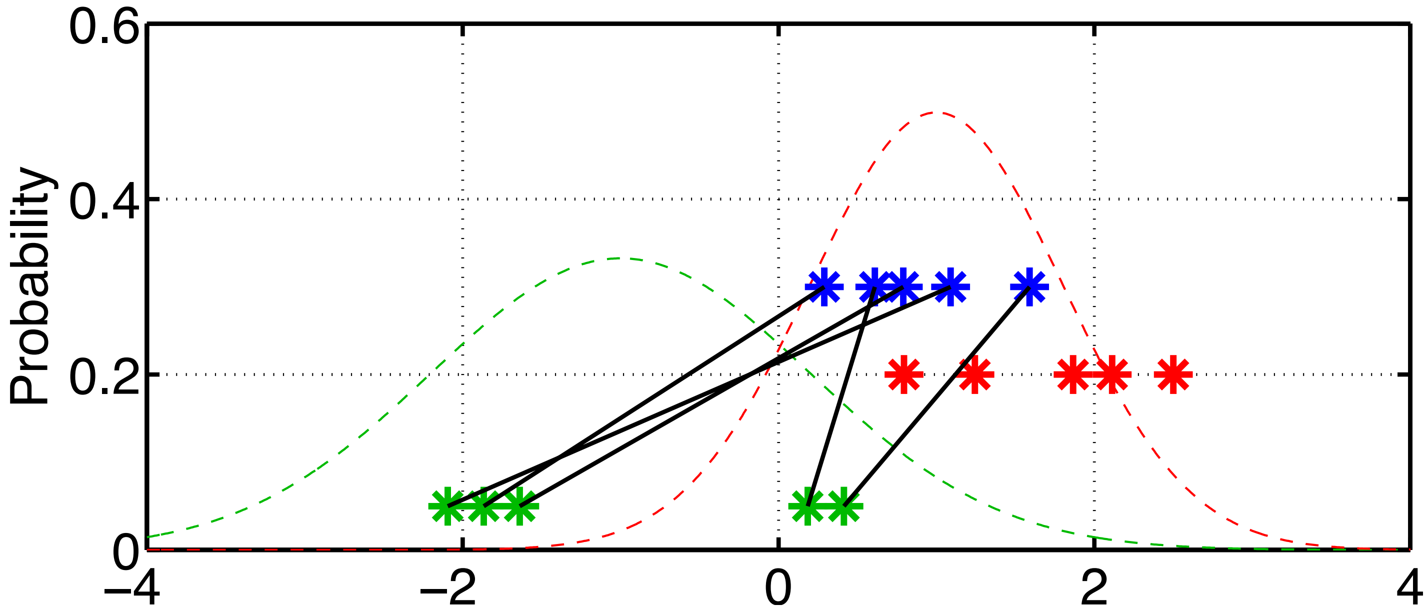
Ensemble Kalman Filter (EnKF)
(*filter_kind*=2 in `&assim_tools_nml`).



Repeat this operation for each joint prior pair.

Ensemble Filter Algorithms:

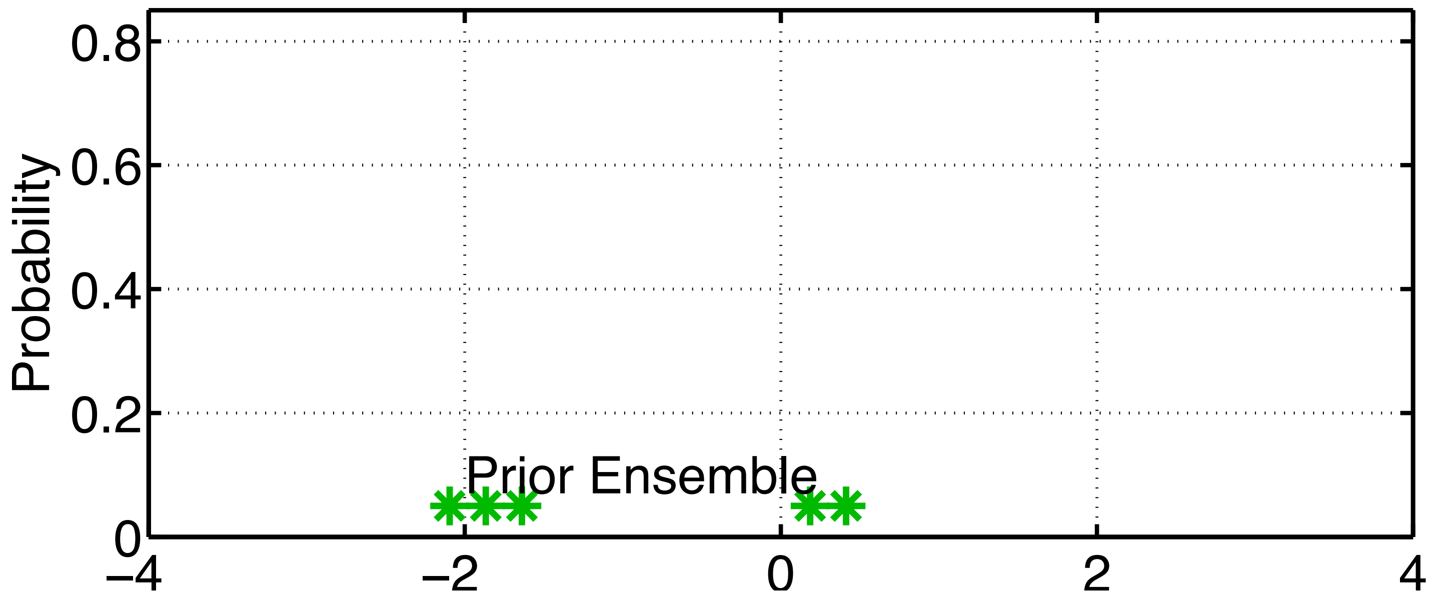
Ensemble Kalman Filter (EnKF)
(*filter_kind*=2 in `&assim_tools_nml`).



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:

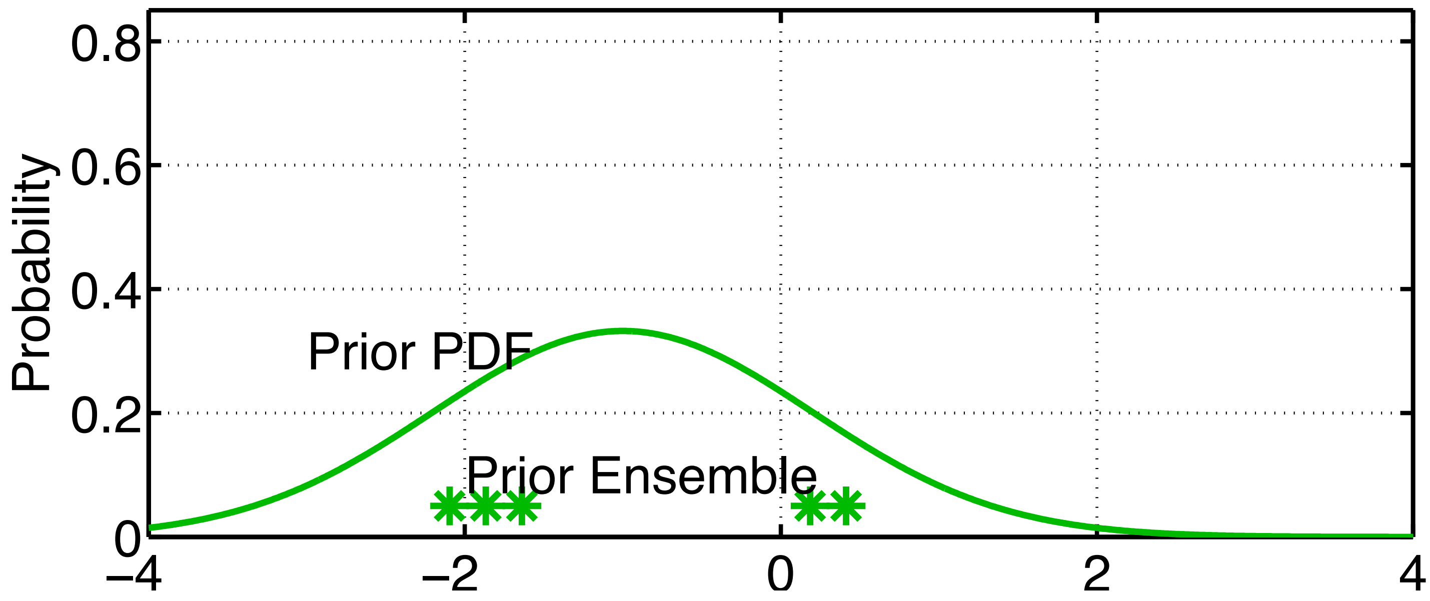
Ensemble Kernel Filter (EKF)
(*filter_kind*=3 in `&assim_tools_nml`).



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`).

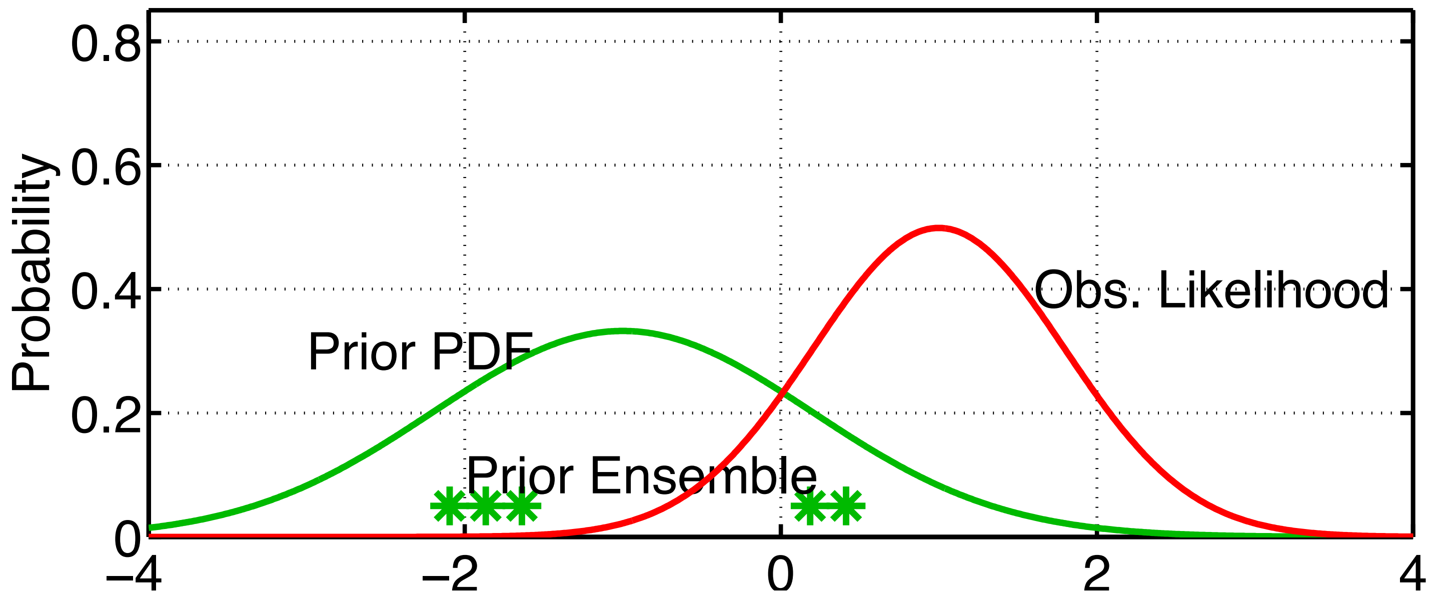


Usually, kernel widths are a function of the sample variance.

Ensemble Filter Algorithms:

Ensemble Kernel Filter (EKF)

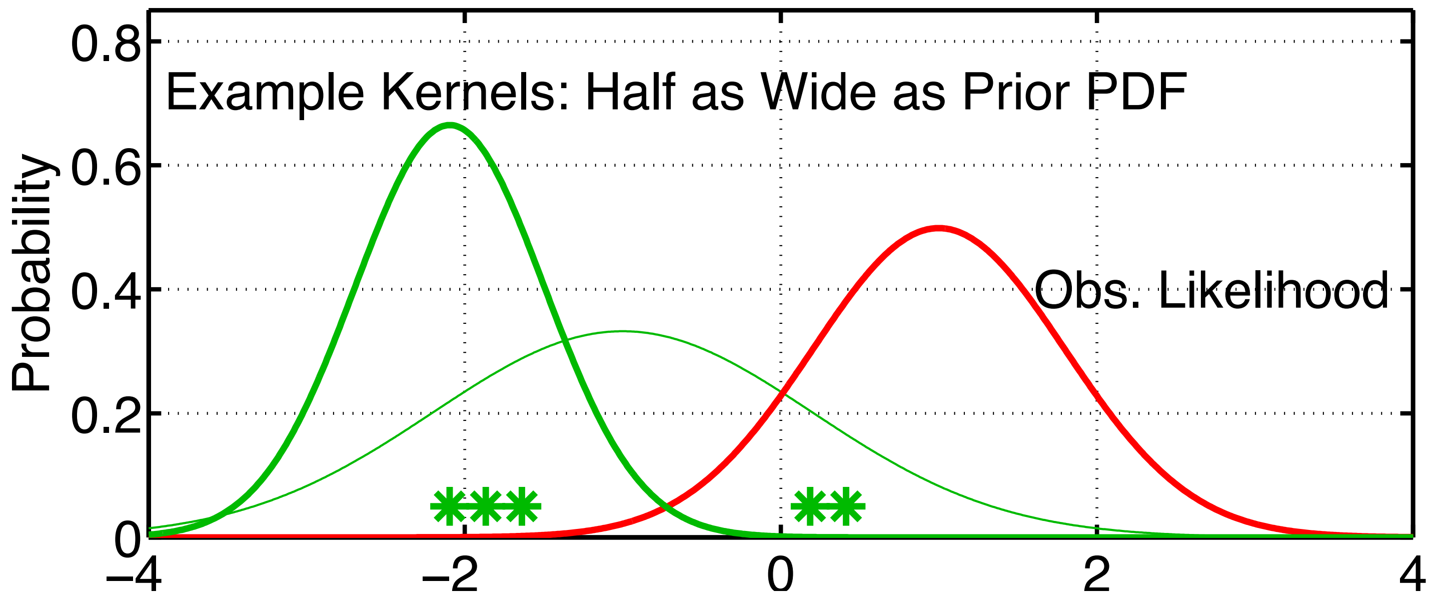
(*filter_kind*=3 in `&assim_tools_nml`).



Usually, kernel widths are a function of the sample variance.

Ensemble Filter Algorithms:

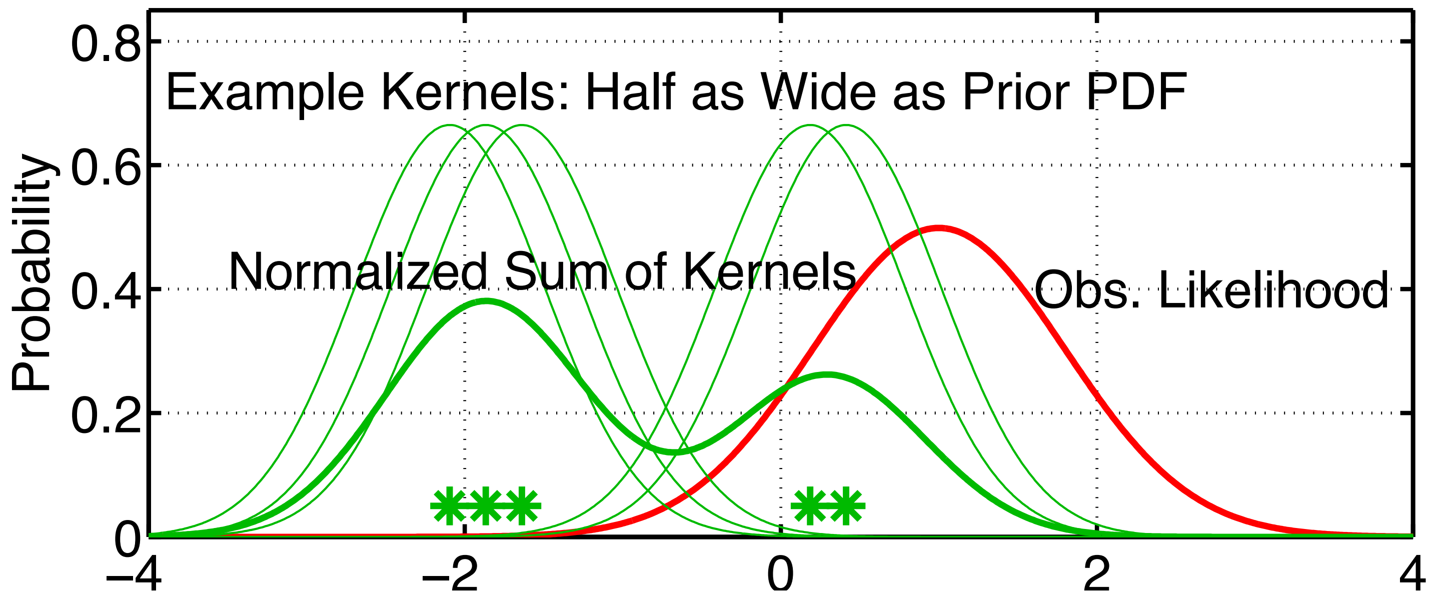
Ensemble Kernel Filter (EKF)
(*filter_kind*=3 in `&assim_tools_nml`).



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

Ensemble Filter Algorithms:

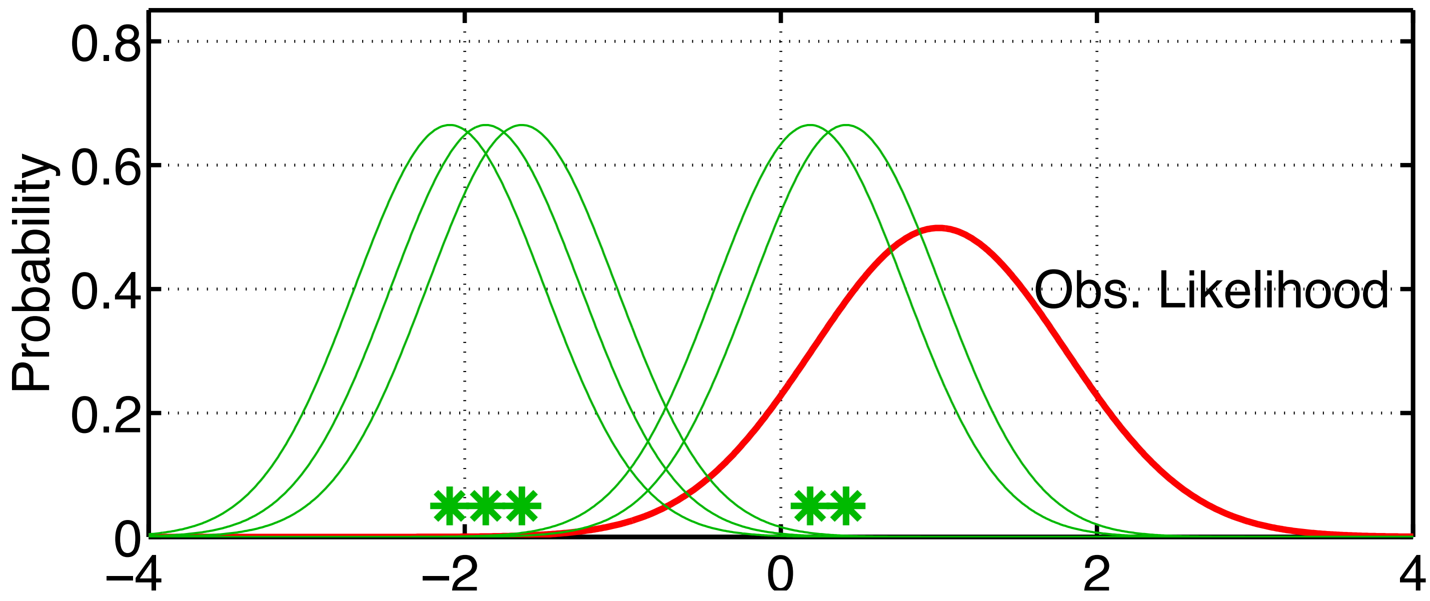
Ensemble Kernel Filter (EKF)
(*filter_kind*=3 in `&assim_tools_nml`).



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

Ensemble Filter Algorithms:

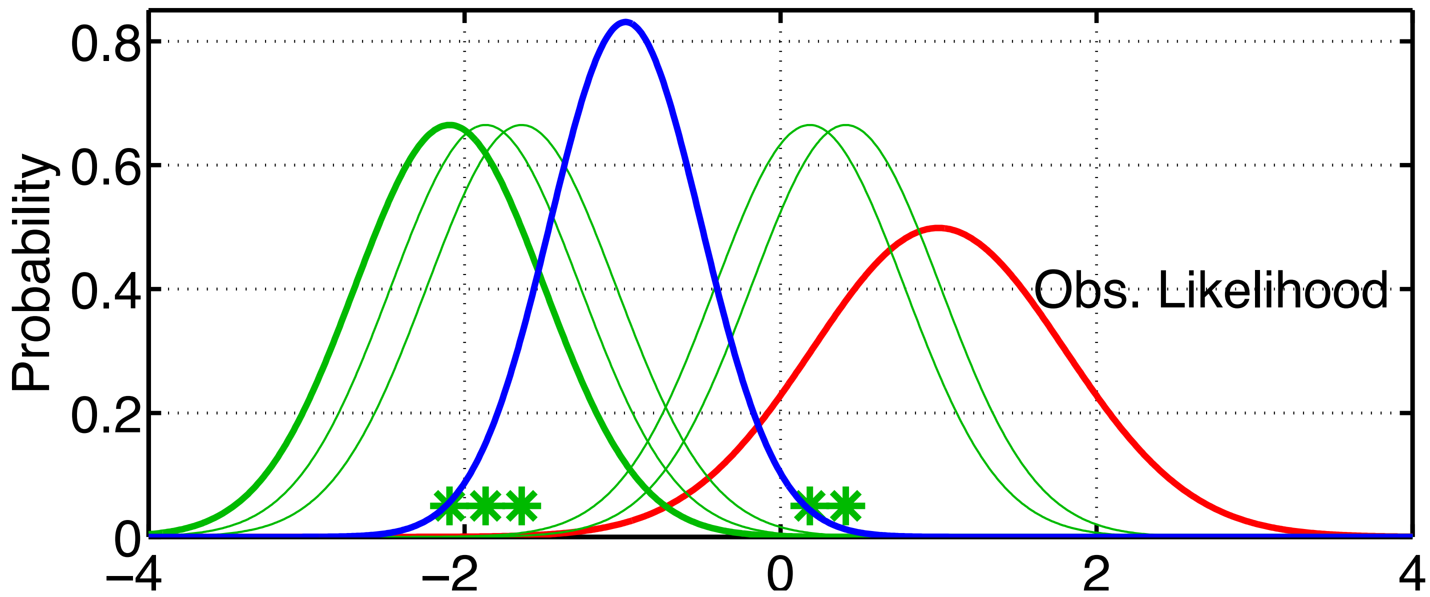
Ensemble Kernel Filter (EKF)
(*filter_kind*=3 in `&assim_tools_nml`).



Apply distributive law to take product: product of the sum is the sum of the products. Otherwise, the product cannot be analytically determined.

Ensemble Filter Algorithms:

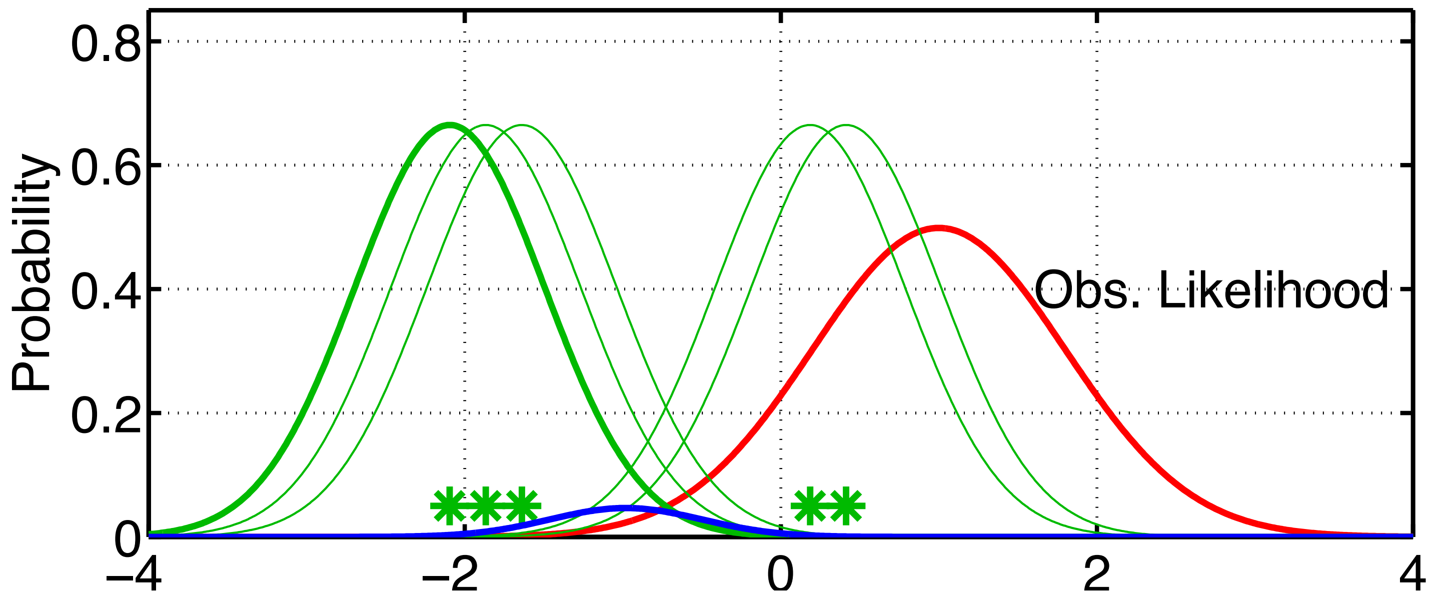
Ensemble Kernel Filter (EKF)
(*filter_kind*=3 in `&assim_tools_nml`).



Compute product of first kernel with Obs. likelihood.

Ensemble Filter Algorithms:

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

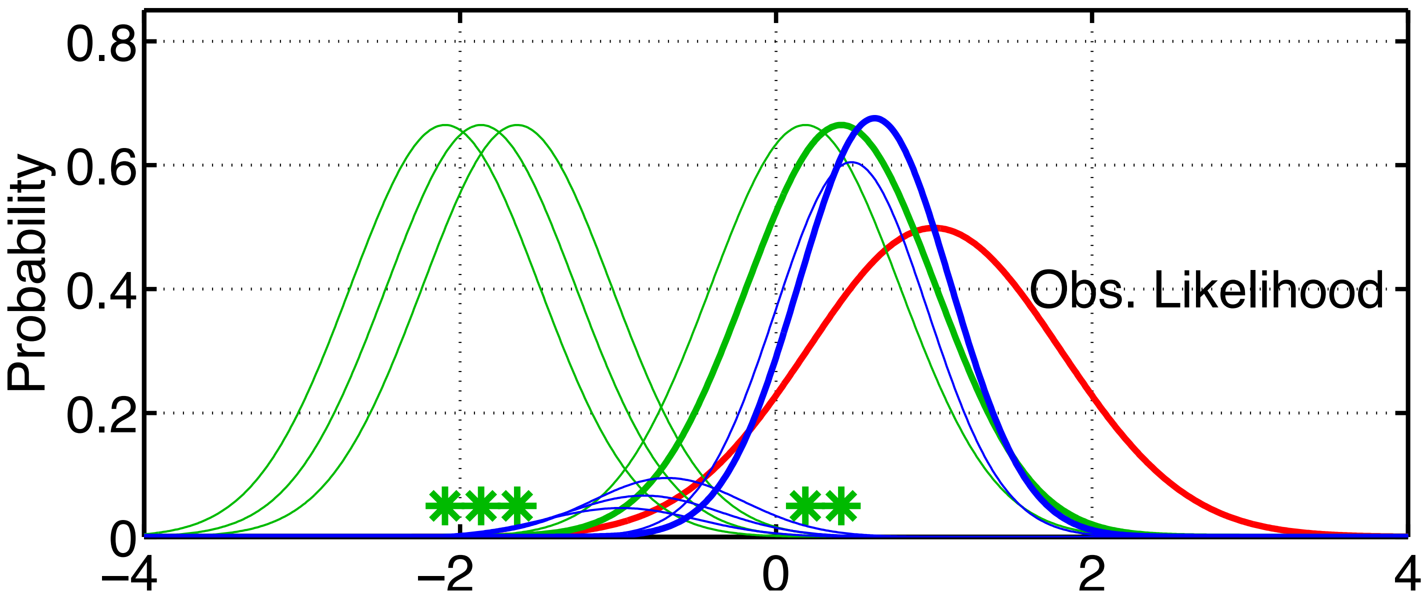


But, can no longer ignore the weight term for product of Gaussians.
Kernels with mean further from observation get less weight.

Ensemble Filter Algorithms:

Ensemble Kernel Filter (EKF)

(*filter_kind*=3 in `&assim_tools_nml`).

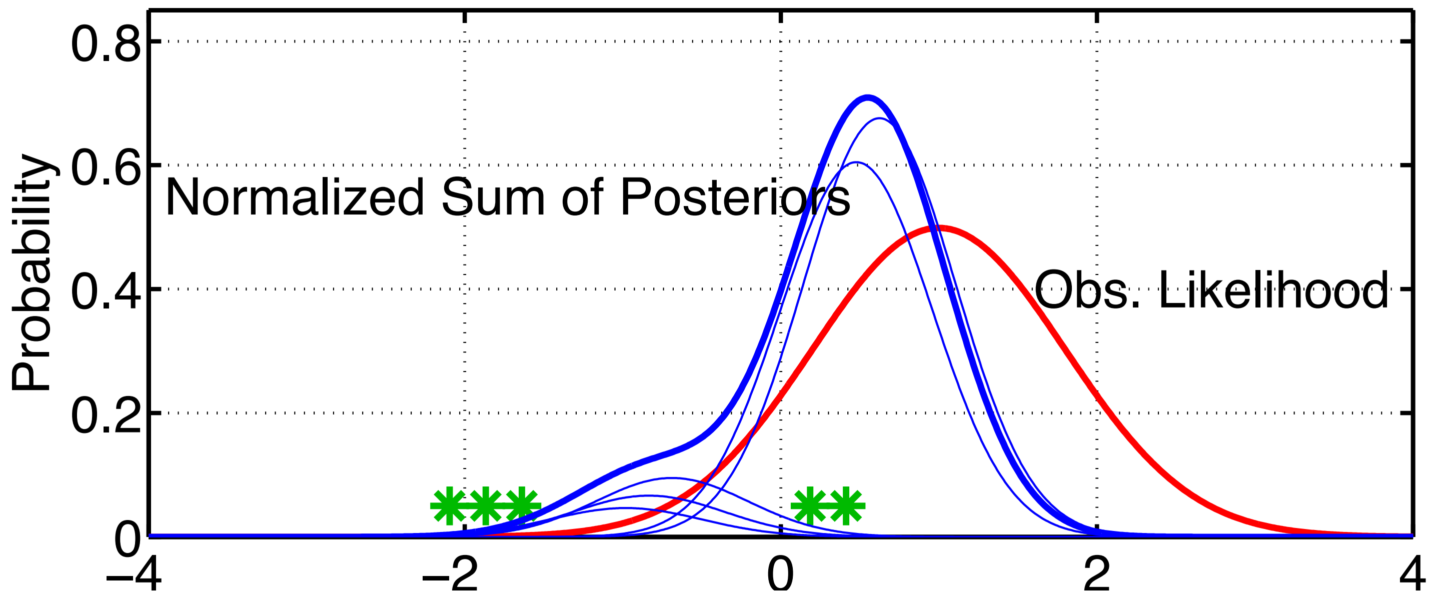


Continue to take products for each kernel in turn.

Closer kernels dominate posterior.

Ensemble Filter Algorithms:

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

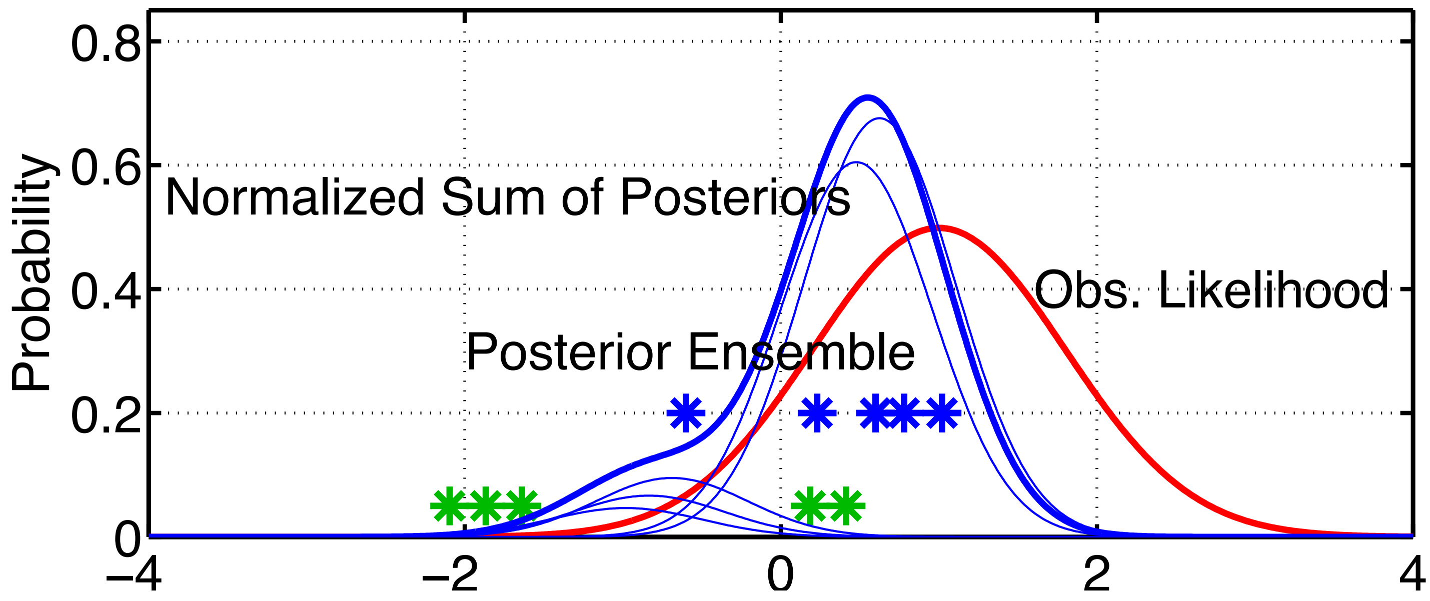


Final posterior is weight-normalized sum of kernel products.

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

Ensemble Filter Algorithms:

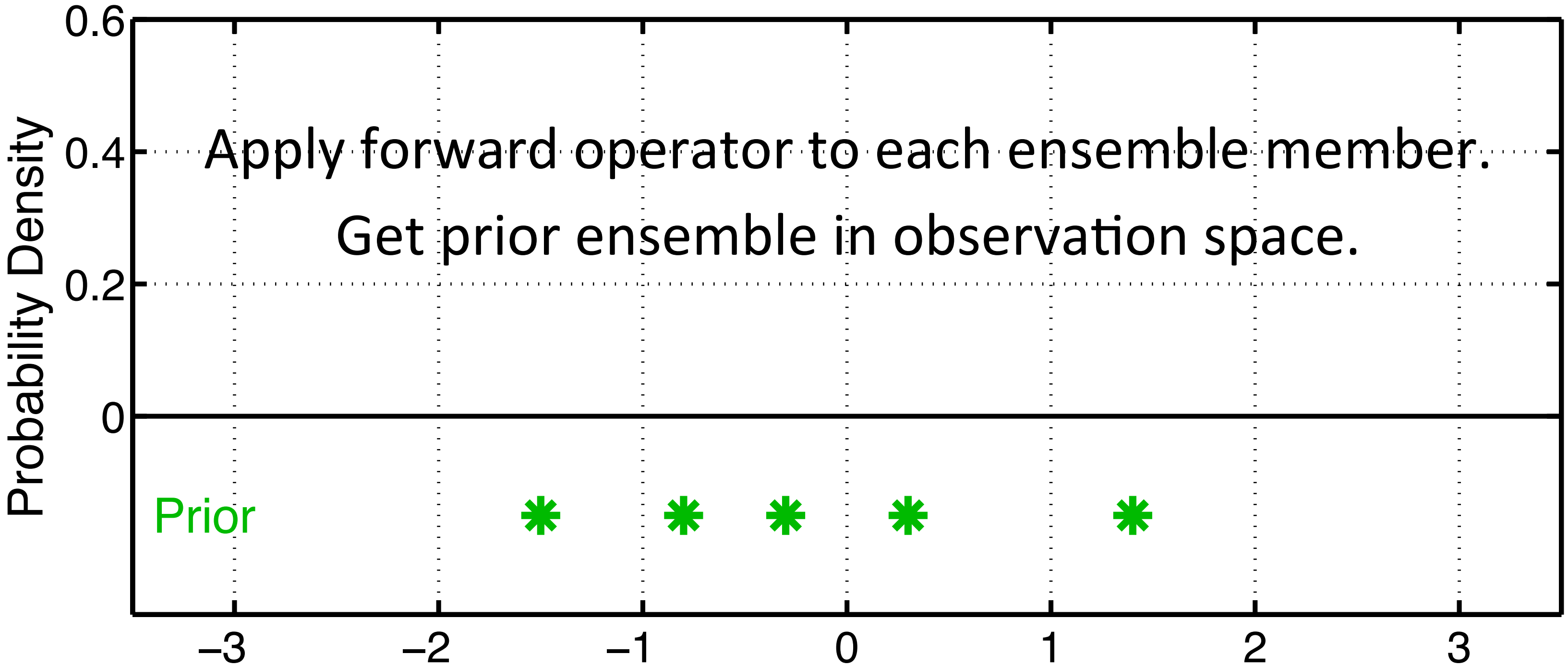
Ensemble Kernel Filter (EKF)
(*filter_kind*=3 in `&assim_tools_nml`).



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:

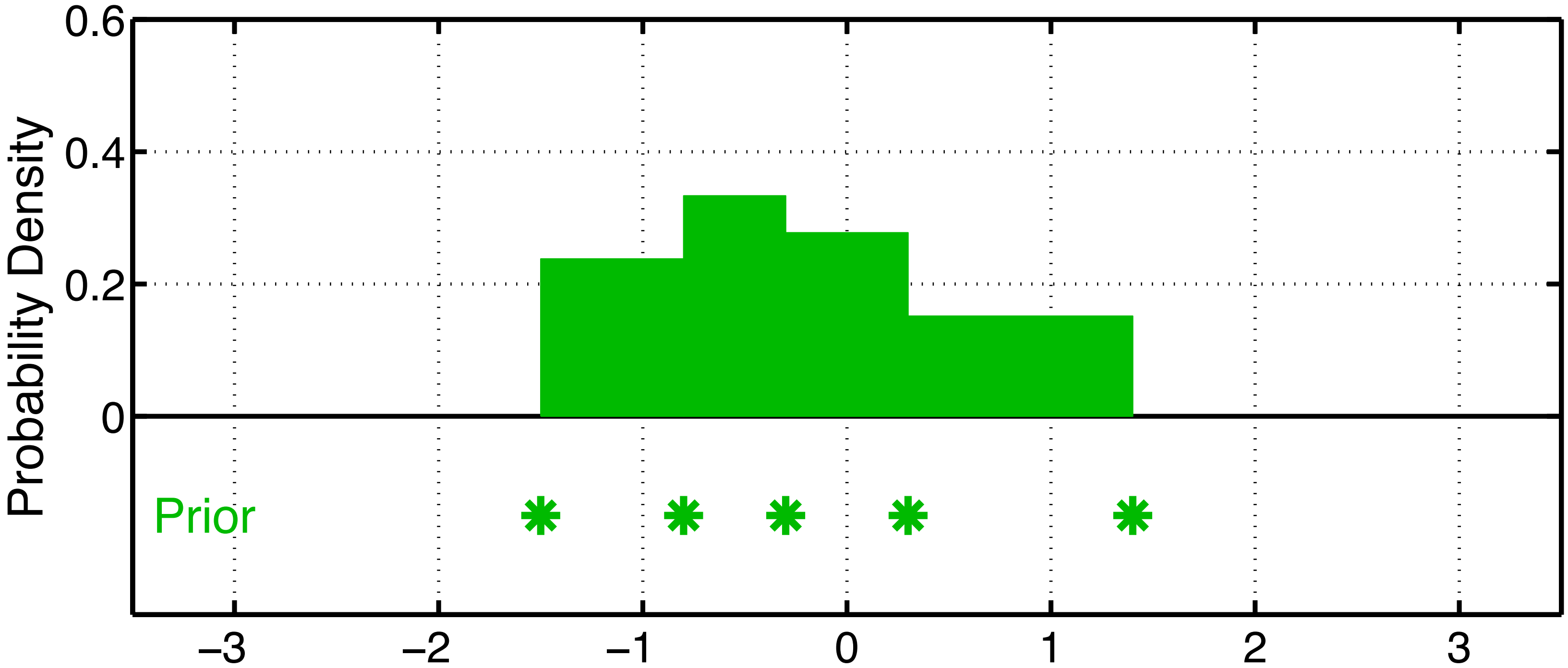
Rank Histogram Filter (*filter_kind*=8 in `&assim_tools_nml`).



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.

Ensemble Filter Algorithms:

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

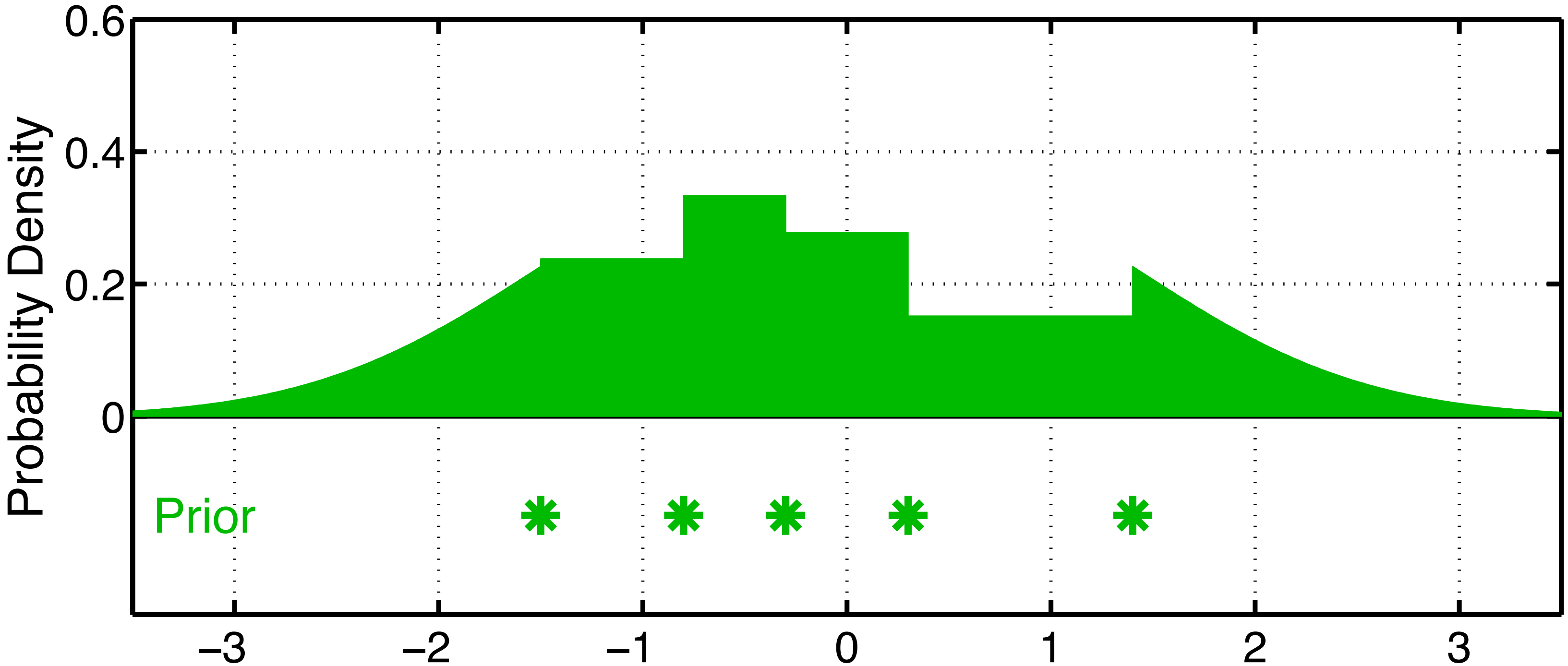


Step 1: Get continuous prior distribution density.

- 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`).

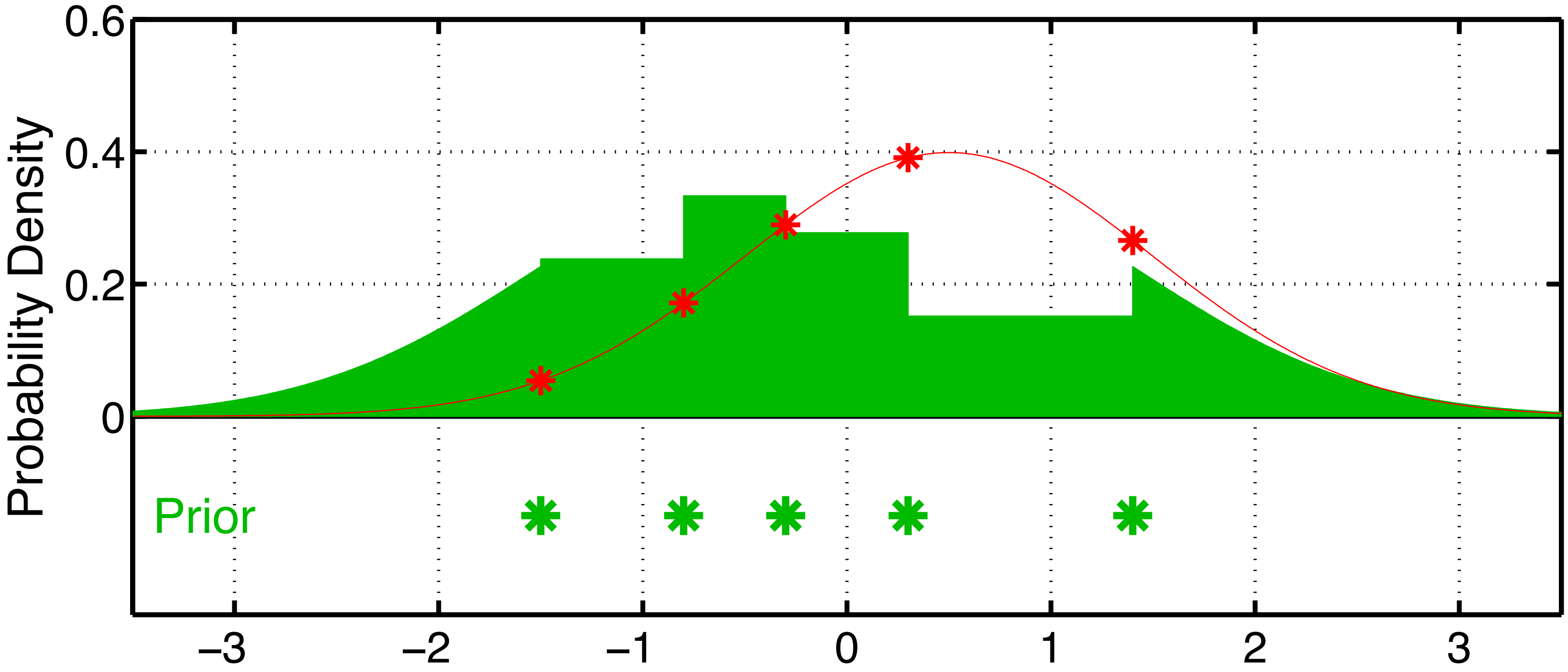


Step 1: Get continuous prior distribution density.

- 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`).

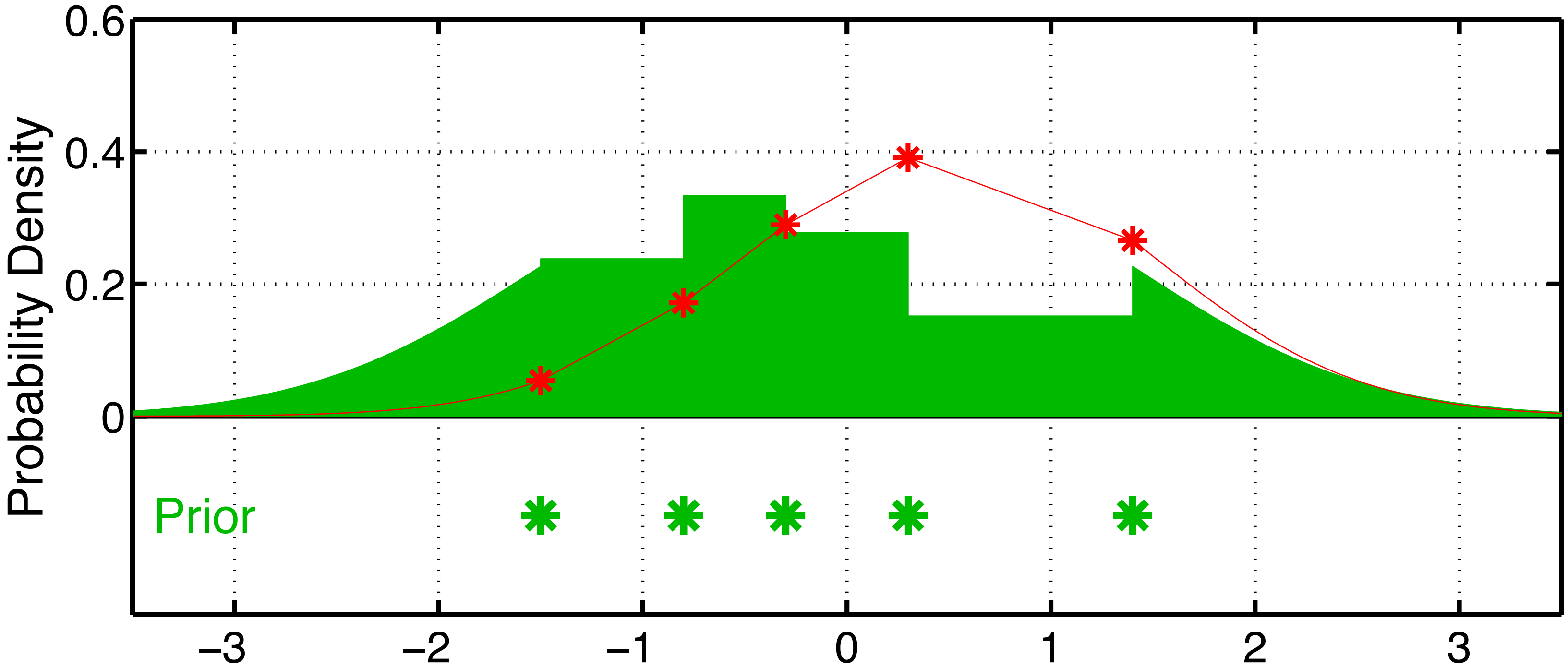


Step 2: Use likelihood to compute weight for each ensemble member.

- 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`).

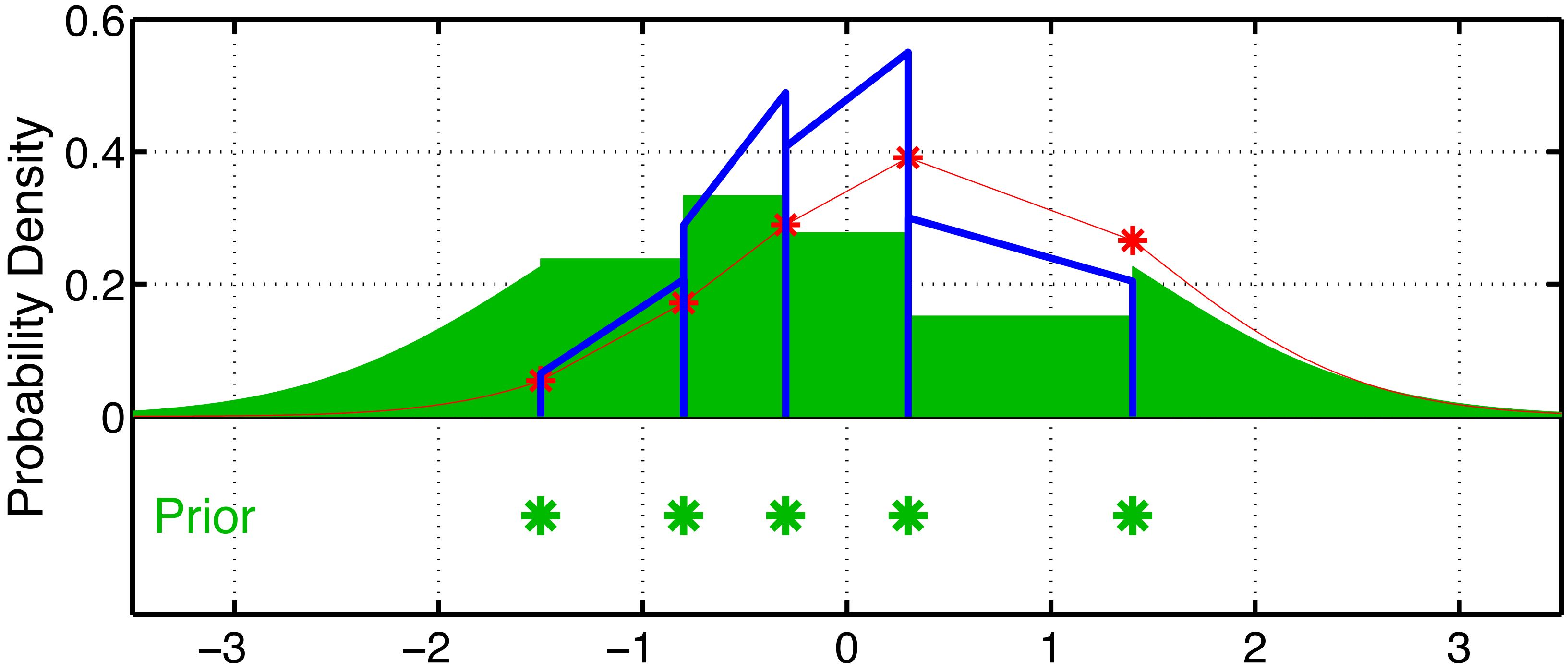


Step 2: Use likelihood to compute weight for each ensemble member.

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

Ensemble Filter Algorithms:

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

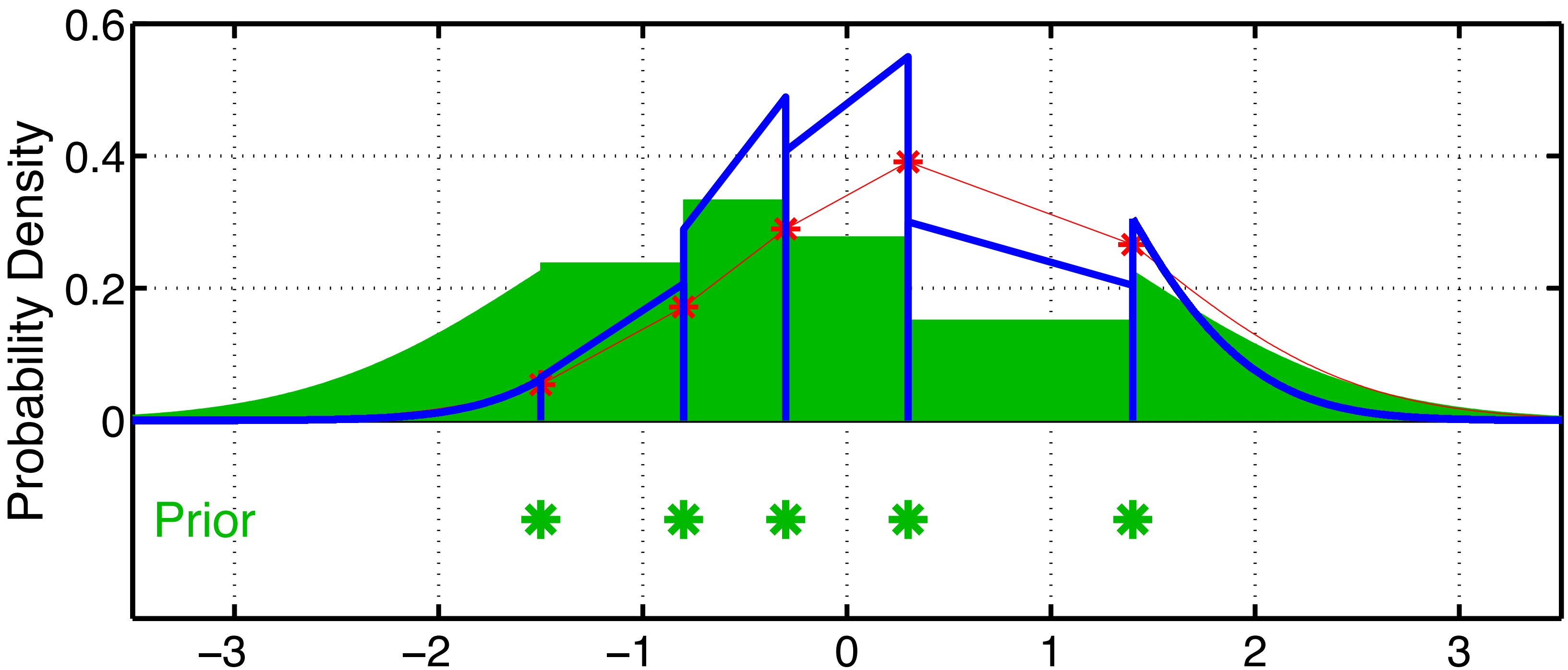


Step 3: Compute continuous posterior distribution.

- 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`).

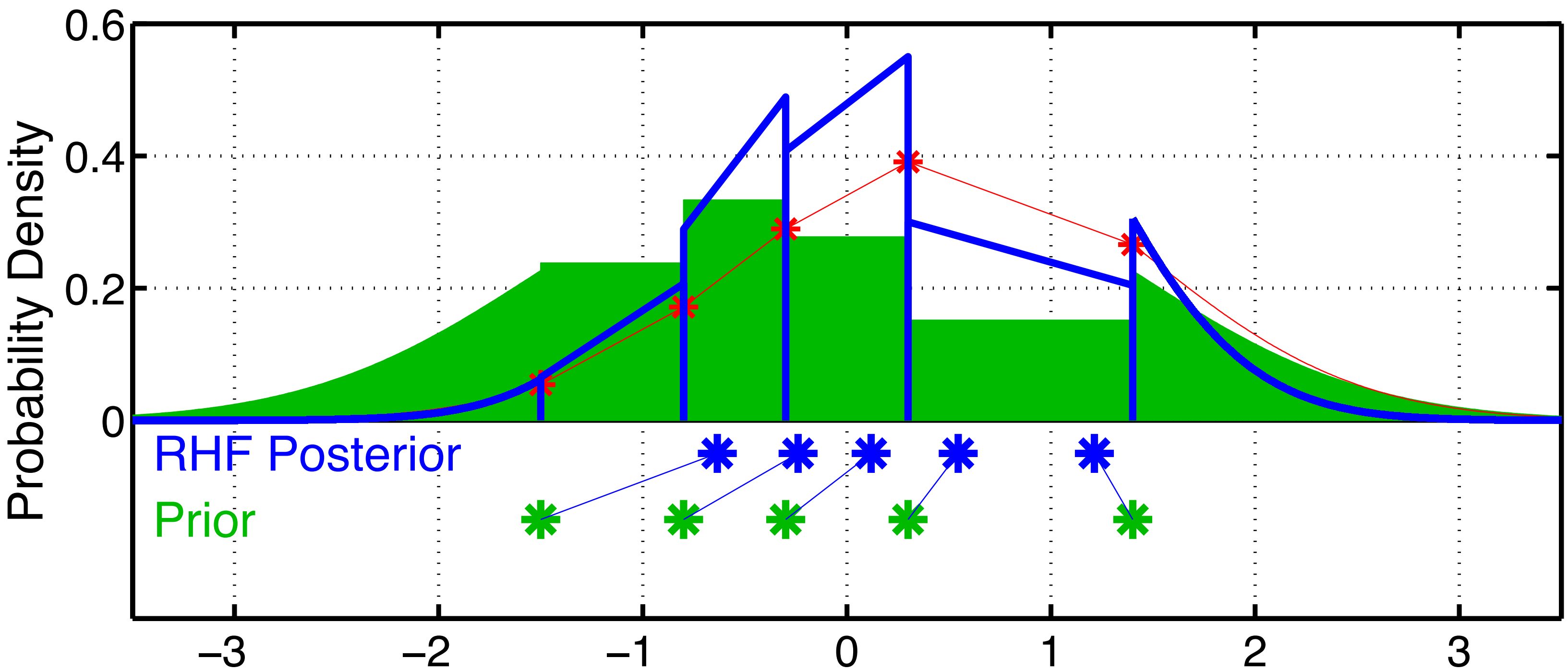


Step 3: Compute continuous posterior distribution.

- Product of prior Gaussian kernel with likelihood for tails.

Ensemble Filter Algorithms:

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

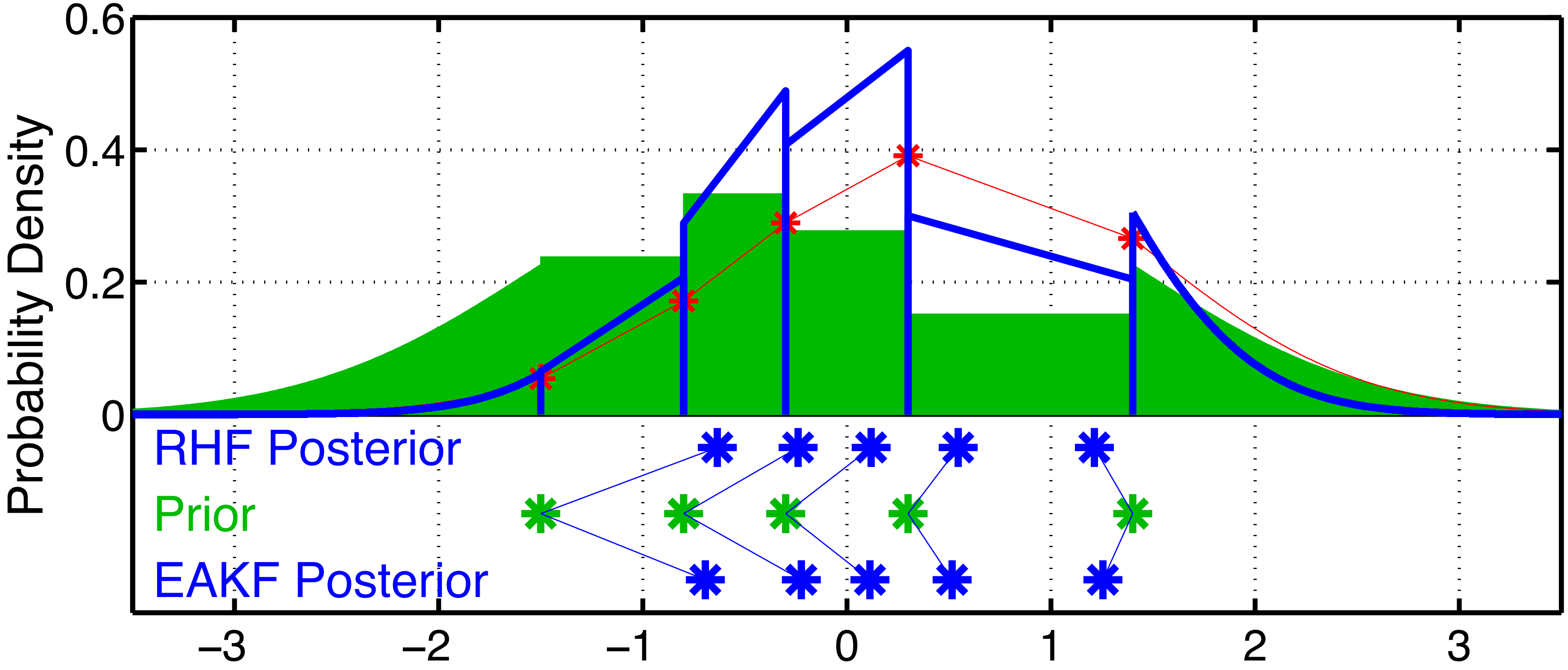


Step 4: Compute updated ensemble members:

- (ens_size +1)$^{-1}$ of posterior mass between each ensemble pair.
- (ens_size +1)$^{-1}$ in each tail.
- Uninformative observation (not shown) would have no impact.

Ensemble Filter Algorithms:

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

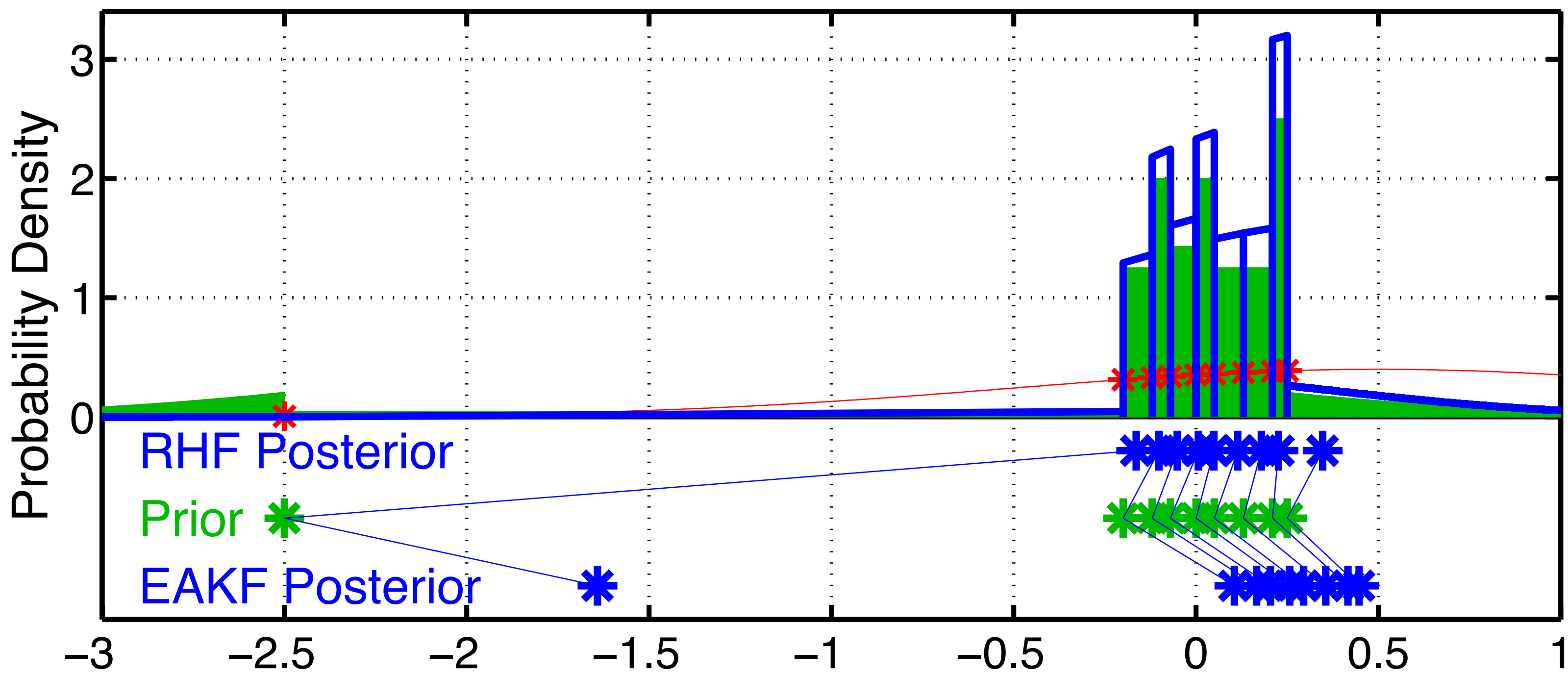


Compare to standard ensemble adjustment Kalman filter (EAKF).

Nearly Gaussian case, differences are small.

Ensemble Filter Algorithms:

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



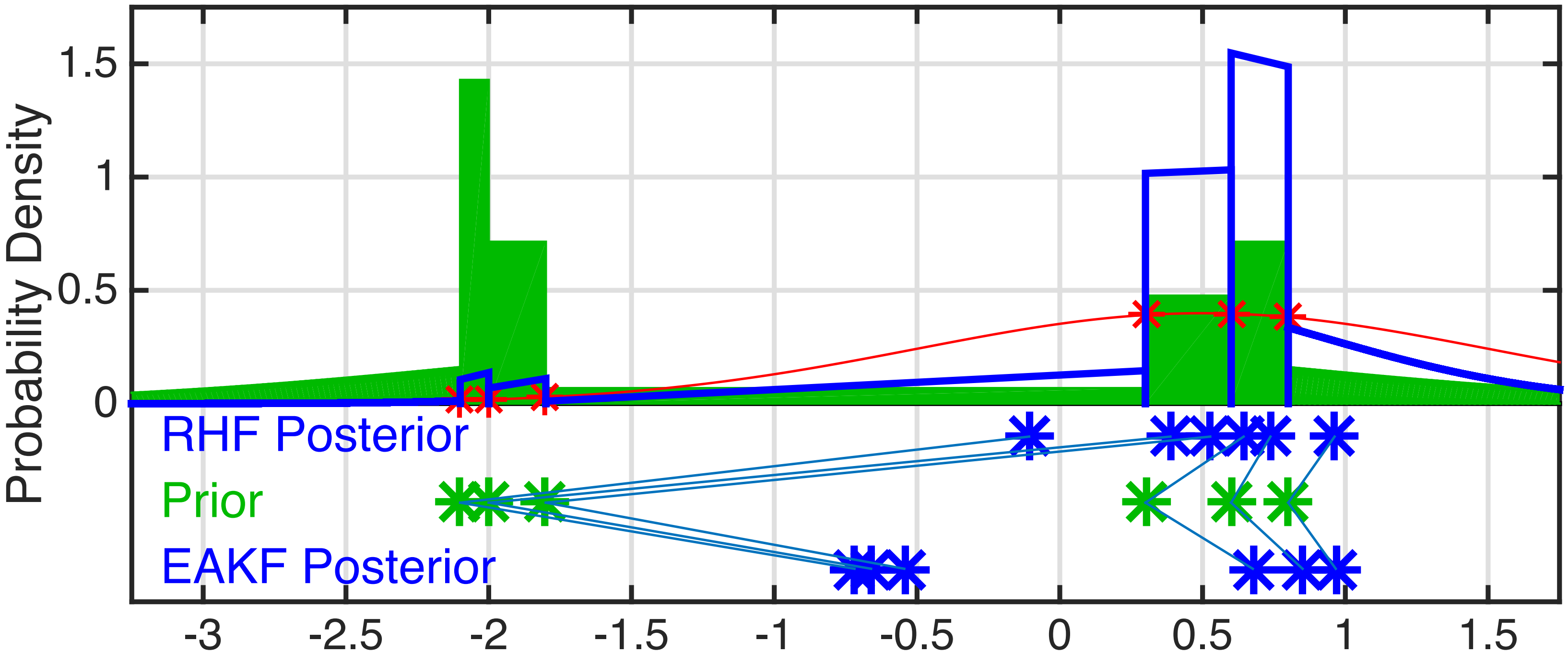
Rank Histogram gets rid of prior outlier that is inconsistent with obs.

EAKF can't get rid of this prior outlier.

Large prior variance from outlier causes EAKF to shift all members too much towards observation (with mean off the page).

Ensemble Filter Algorithms:

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



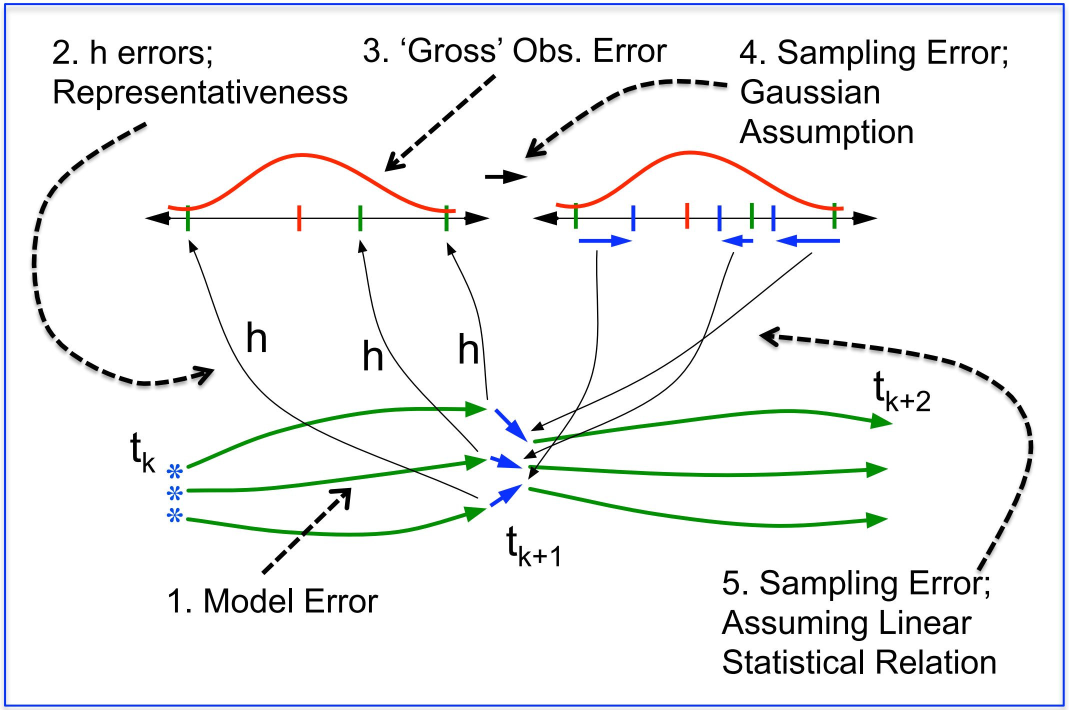
Convective-scale models (and land models) have analogous behavior.

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.

Dealing with Ensemble Filter Errors



Fix 1, 2, 3 independently, HARD but ongoing.

Often, ensemble filters...

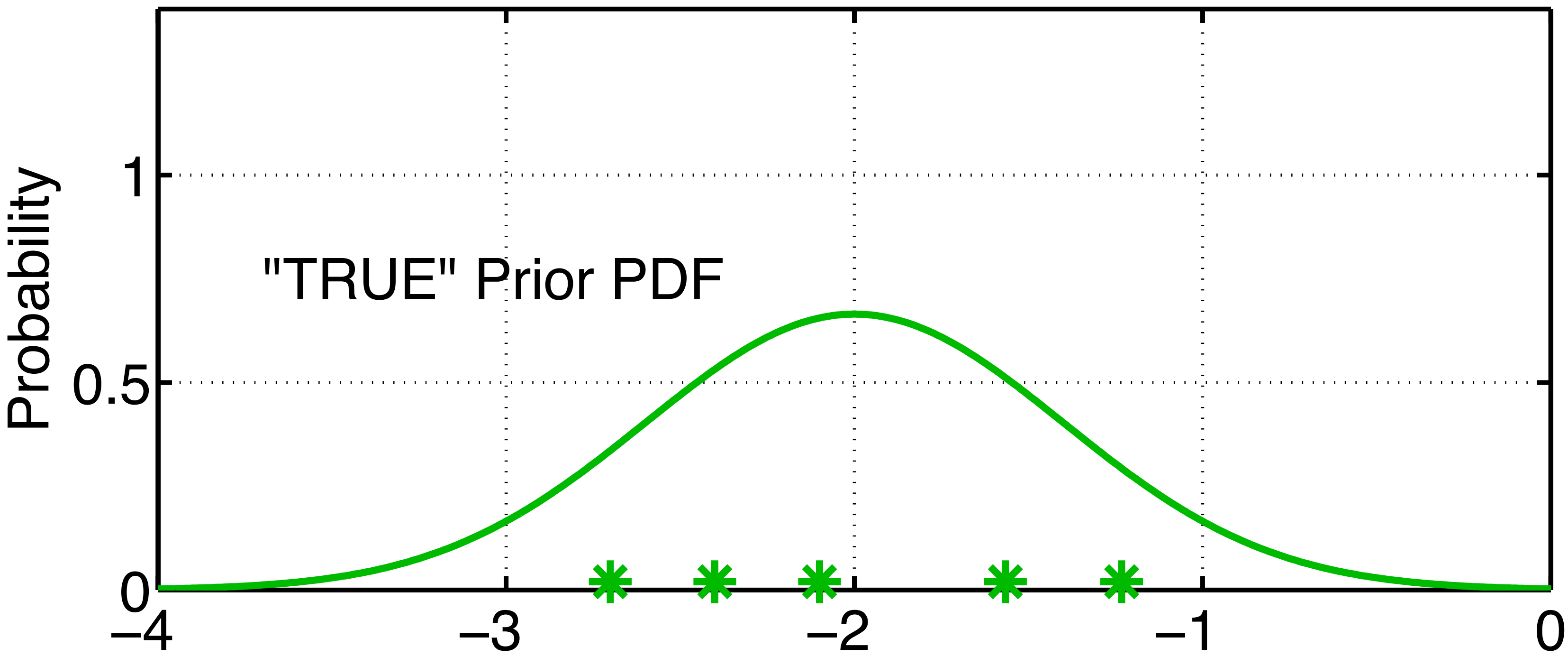
1-4: Variance inflation, Increase prior uncertainty to give obs more impact.

5. 'Localization': only let obs. impact a set of 'nearby' state variables.

Often smoothly decrease impact to 0 as function of distance.

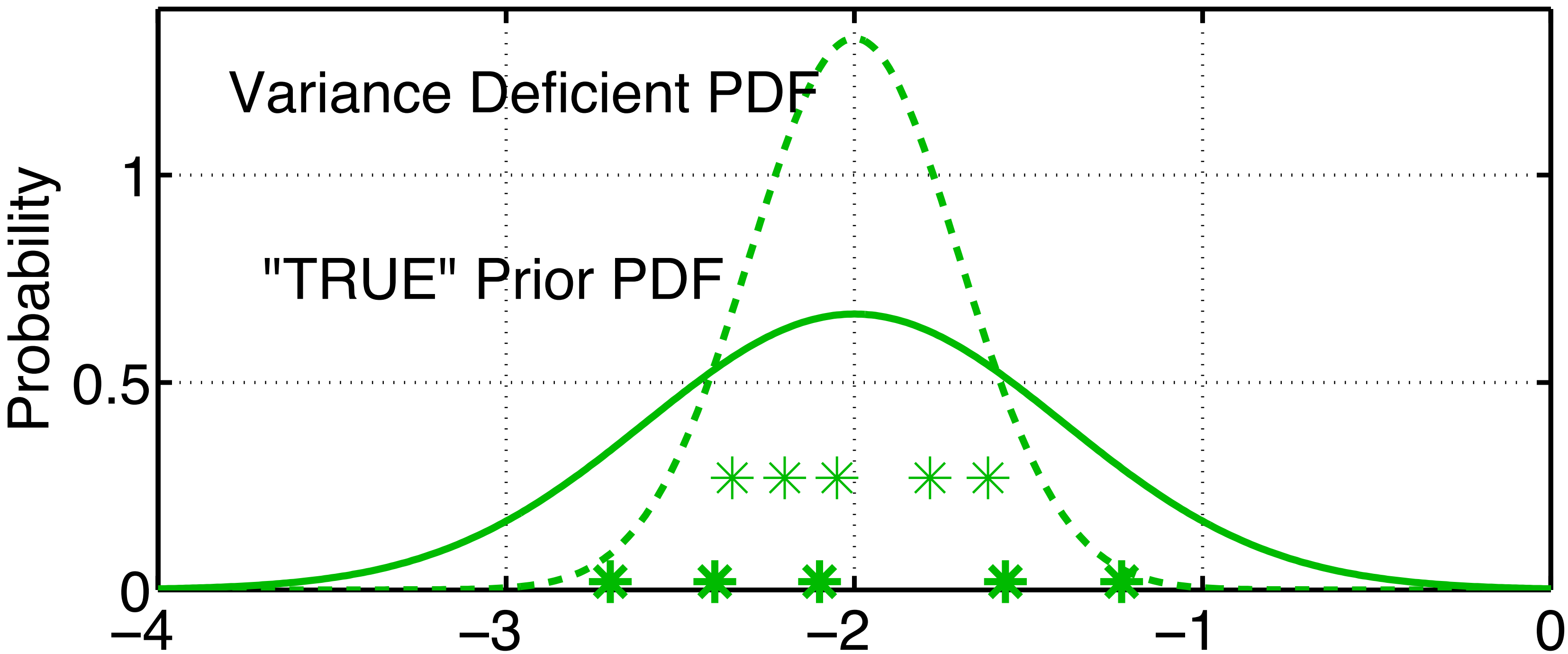
Model/Filter Error: Filter Divergence and Variance Inflation

1. History of observations and physical system => ‘true’ distribution.



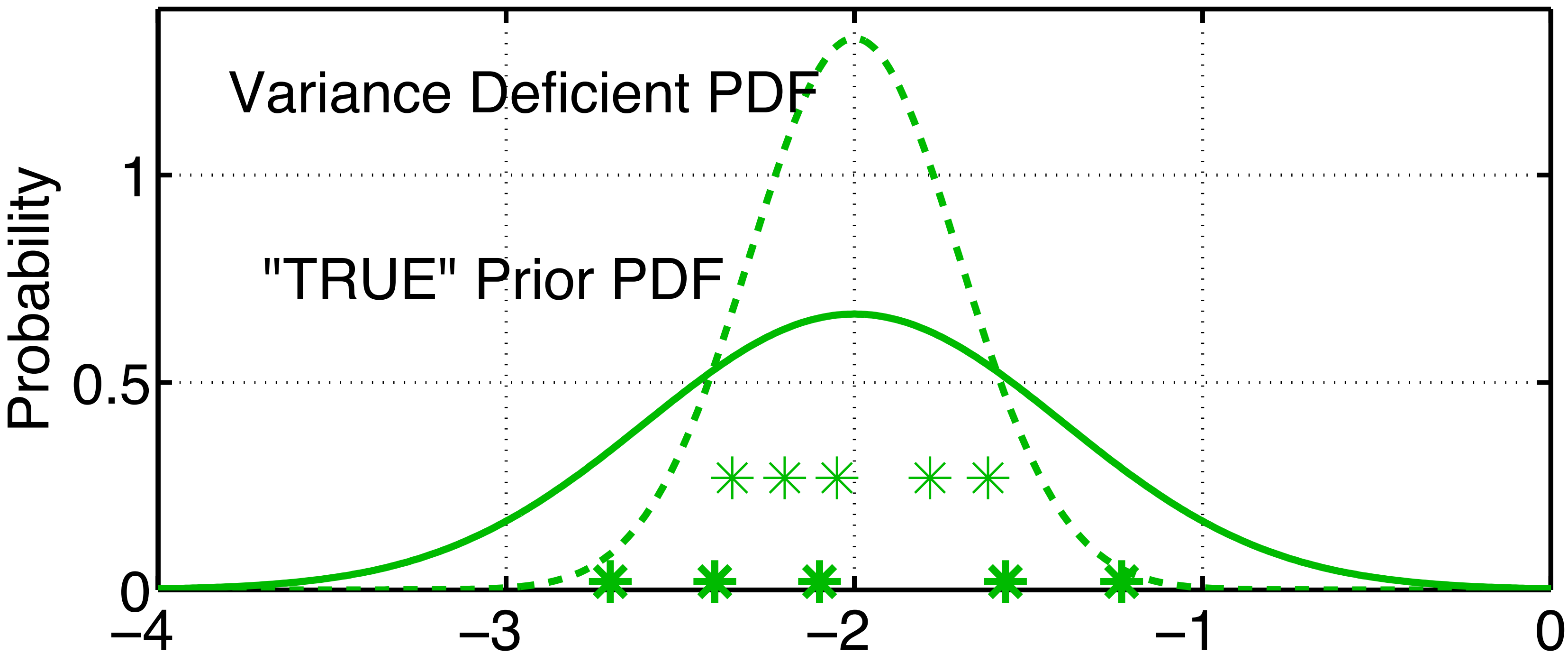
Model/Filter Error: Filter Divergence and Variance Inflation

1. History of observations and physical system => 'true' distribution.
2. Sampling error, some model errors lead to insufficient prior variance.
3. Can lead to 'filter divergence': prior is too confident, obs. Ignored.



Model/Filter Error: Filter Divergence and Variance Inflation

1. History of observations and physical system => 'true' distribution.
2. Sampling error, some model errors lead to insufficient prior variance.
3. Can lead to 'filter divergence': prior is too confident, obs. Ignored.

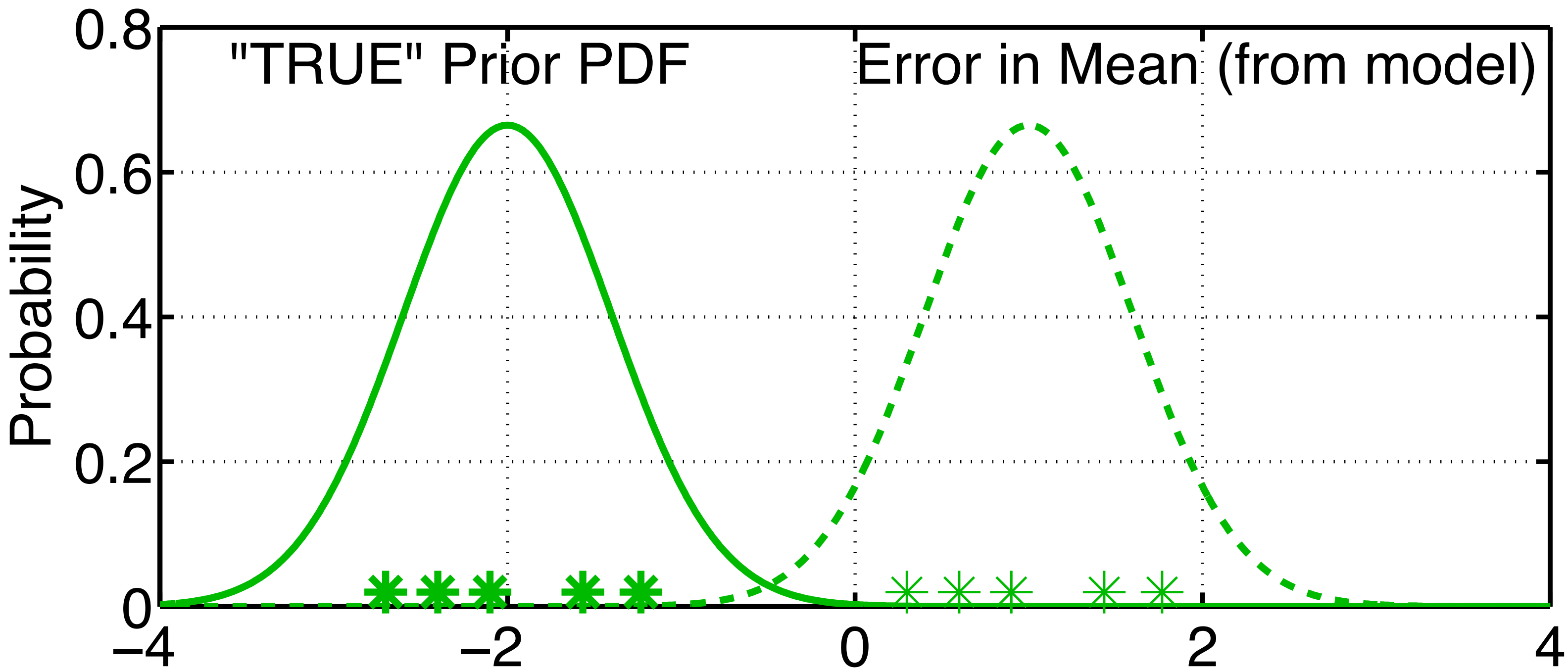


Naïve solution is variance inflation: just increase spread of prior.

For ensemble member i, $inflate(x_i) = \sqrt{\lambda}\left(x_i - \overline{x}\right) + \overline{x}$

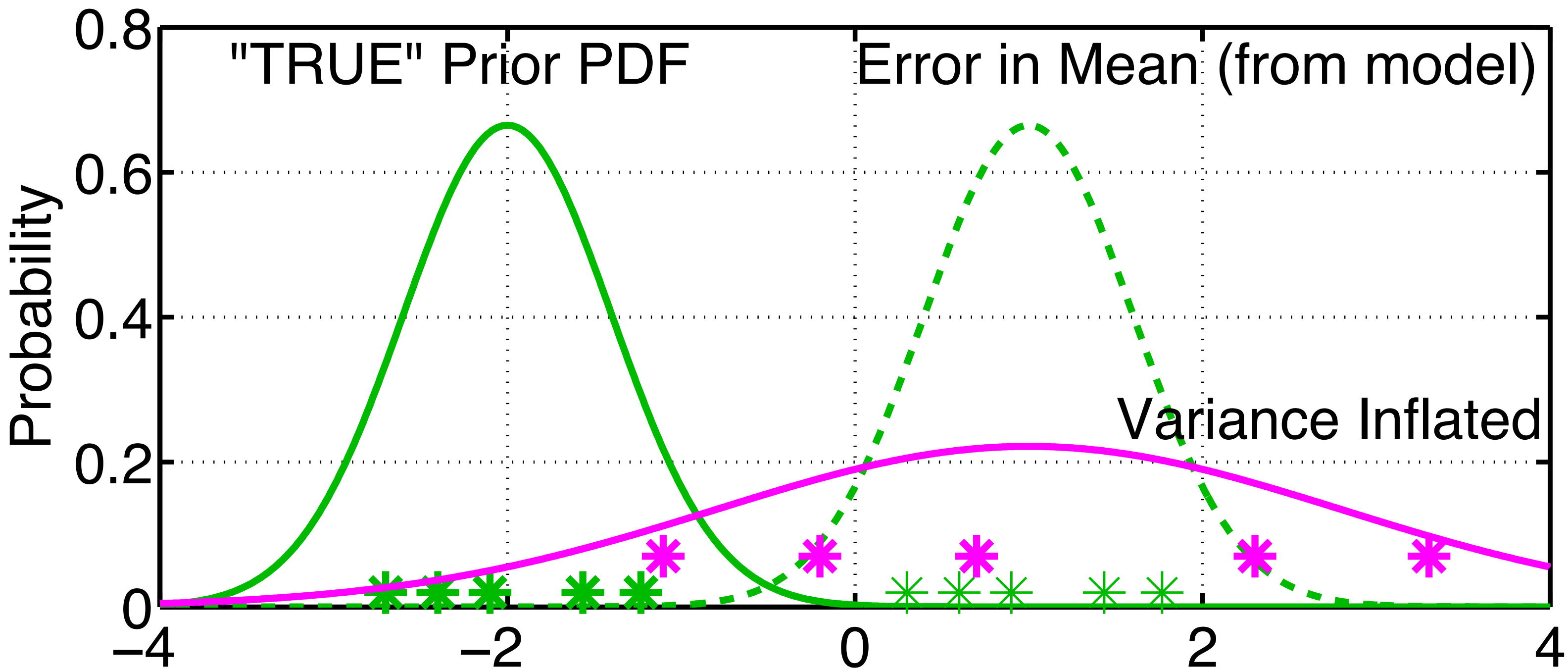
Model/Filter Error: Filter Divergence and Variance Inflation

1. History of observations and physical system => ‘true’ distribution.
2. Most model errors also lead to erroneous shift in entire distribution.
3. Again, prior can be viewed as being TOO CERTAIN.



Model/Filter Error: Filter Divergence and Variance Inflation

1. History of observations and physical system => ‘true’ distribution.
2. Most model errors also lead to erroneous shift in entire distribution.
3. Again, prior can be viewed as being TOO CERTAIN.



Inflating can ameliorate this.
Obviously, if we knew E(error), we’d correct for it directly.

Physical Space Variance Inflation

Inflate all state variables by same amount before assimilation.

Capabilities:

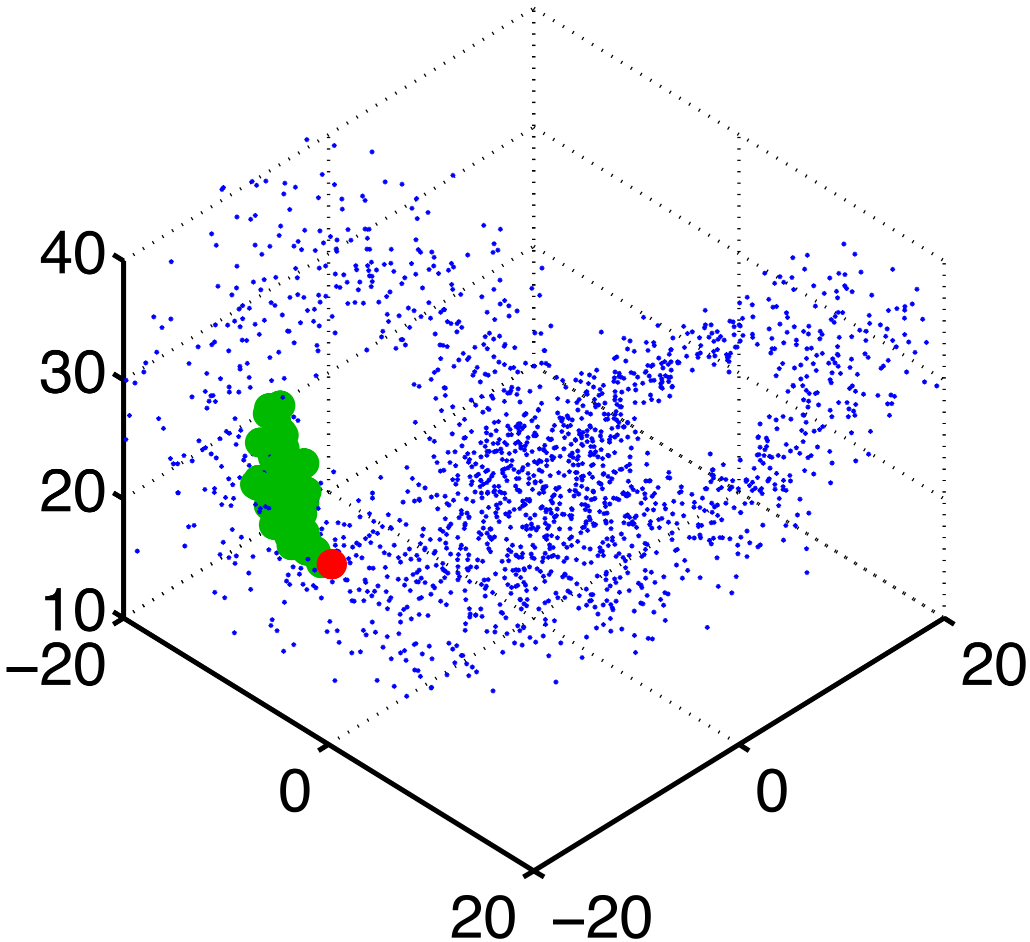
1. Can be effective for a variety of models.
2. Can maintain linear balances.
3. Prior continues to resemble that from the first guess.
4. Simple and computationally cheap.

Liabilities:

1. State variables not constrained by observations can 'blow up'. For instance unobserved regions near the top of AGCMs.
2. Magnitude of λ normally selected by trial and error.

Physical Space Variance Inflation in Lorenz 63

Observation outside prior: danger of filter divergence.

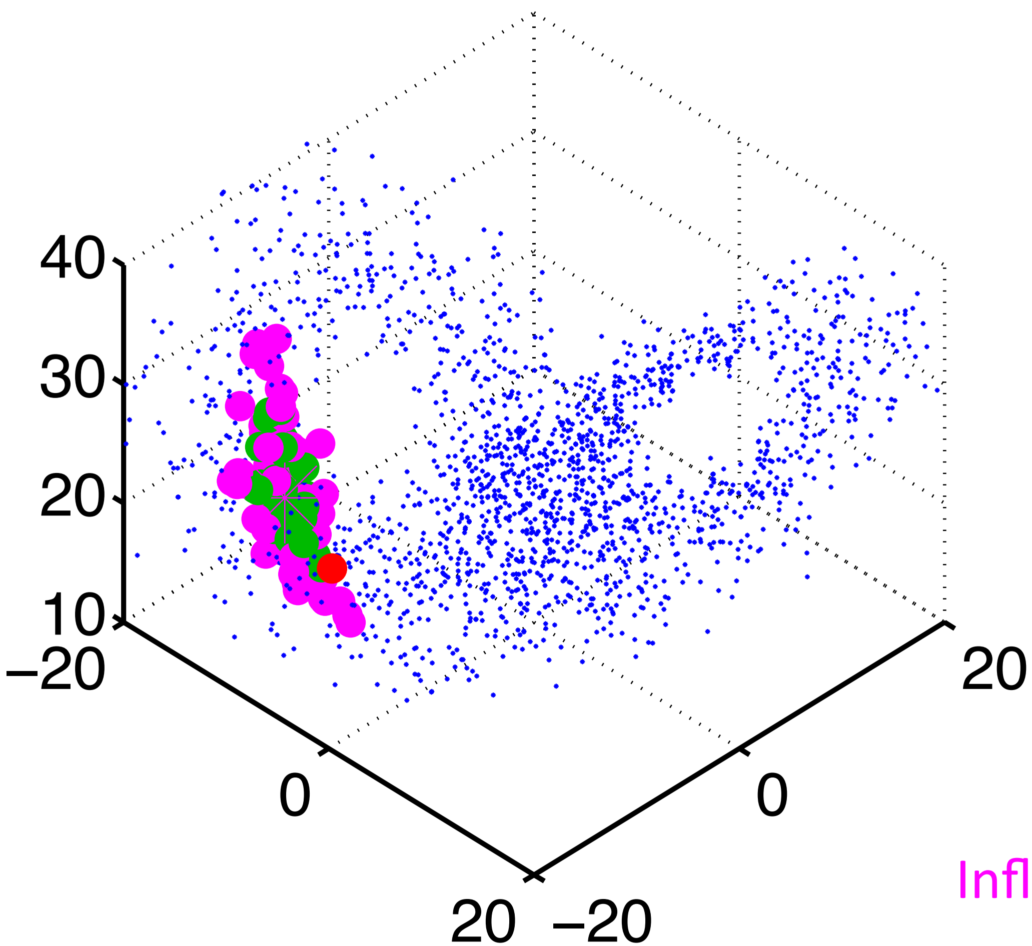


Observation in red.

Prior ensemble in green.

Physical Space Variance Inflation in Lorenz 63

After inflating, observation is in prior cloud: filter divergence avoided.



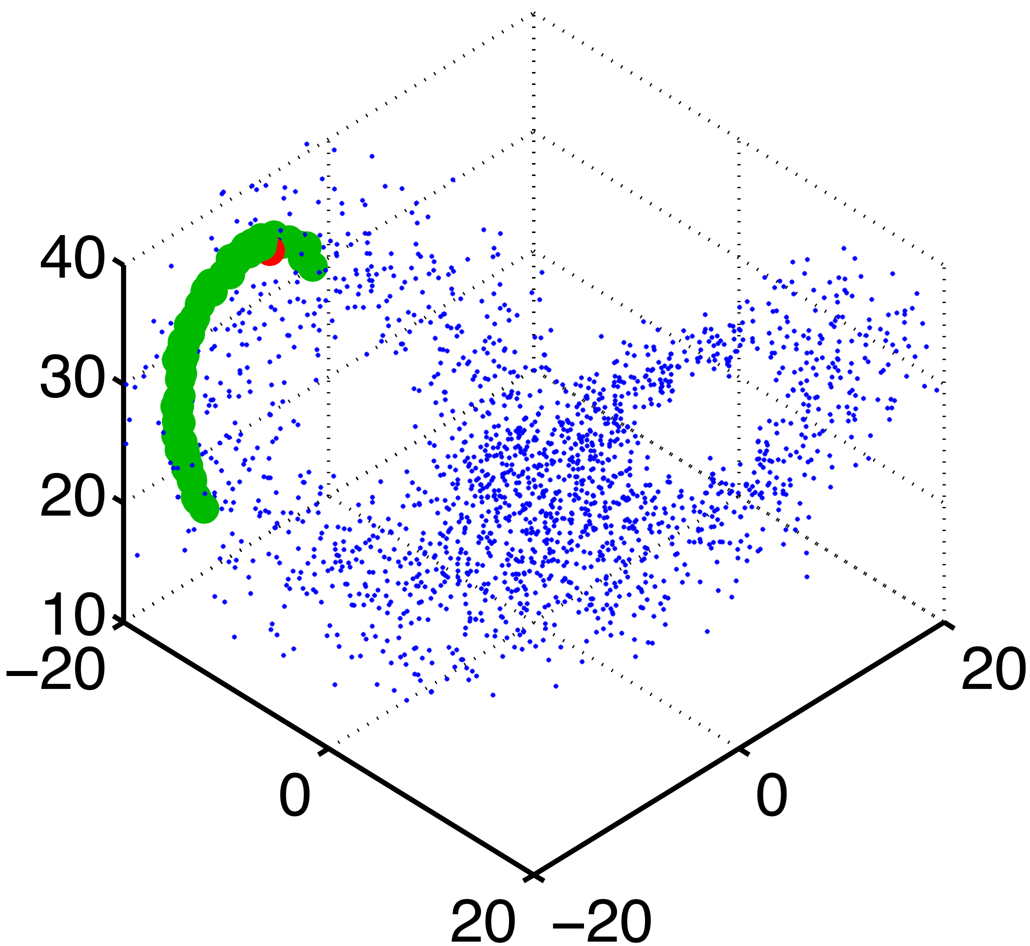
Observation in red.

Prior ensemble in green.

Inflated ensemble in magenta.

Physical Space Variance Inflation in Lorenz 63

Prior distribution is significantly 'curved'.

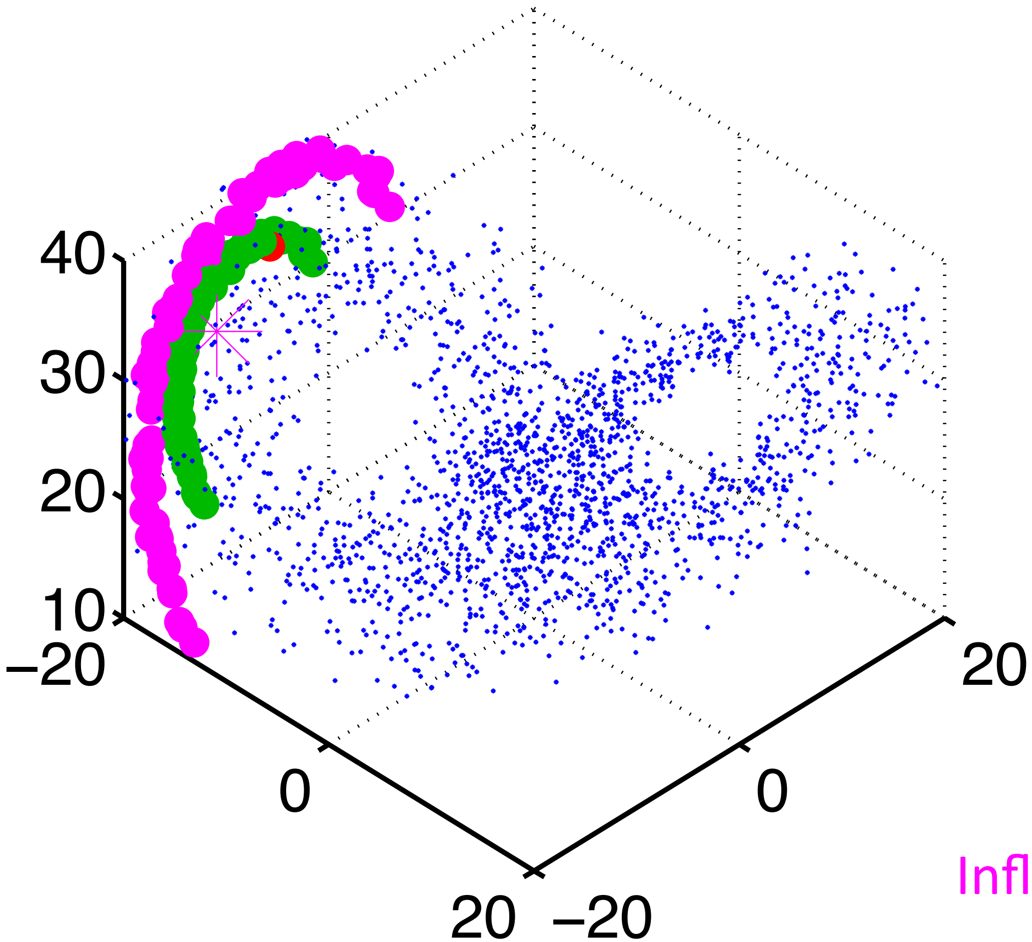


Observation in red.

Prior ensemble in green.

Physical Space Variance Inflation in Lorenz 63

Inflated prior outside attractor. Posterior will also be off attractor.



Can lead to transient off-attractor behavior or…

model ‘blow-up’.

Observation in red.

Prior ensemble in green.

Inflated ensemble in magenta.