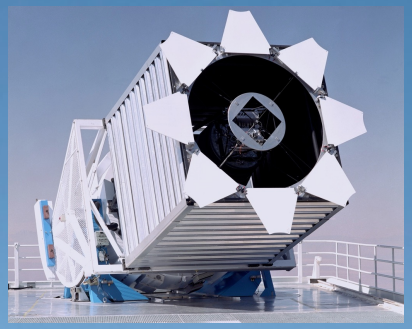
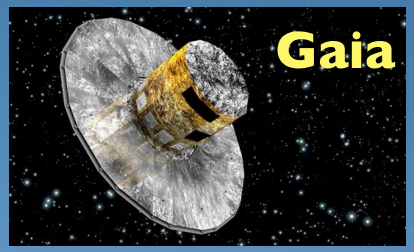
Week 9: Density estimation

Željko Ivezić and Mario Jurić, Department of Astronomy, UW

LSST



SDSS



Density Estimation

Very often we encounter problems where data are samples taken from a continuous distribution.

For example:

- The distribution of mass in the Galaxy can be considered as smooth and continuous at large scales; the stars we individually measure sample it.
  - Another way of looking at this is that the stars are drawn from an underlying PDF.
- The distribution of galaxies in the universe is similarly continuous at large scales
- We can also look at distributions in parameter spaces: e.g. the distribution of stars in (g-r, r) color-magnitude diagram.

Outline

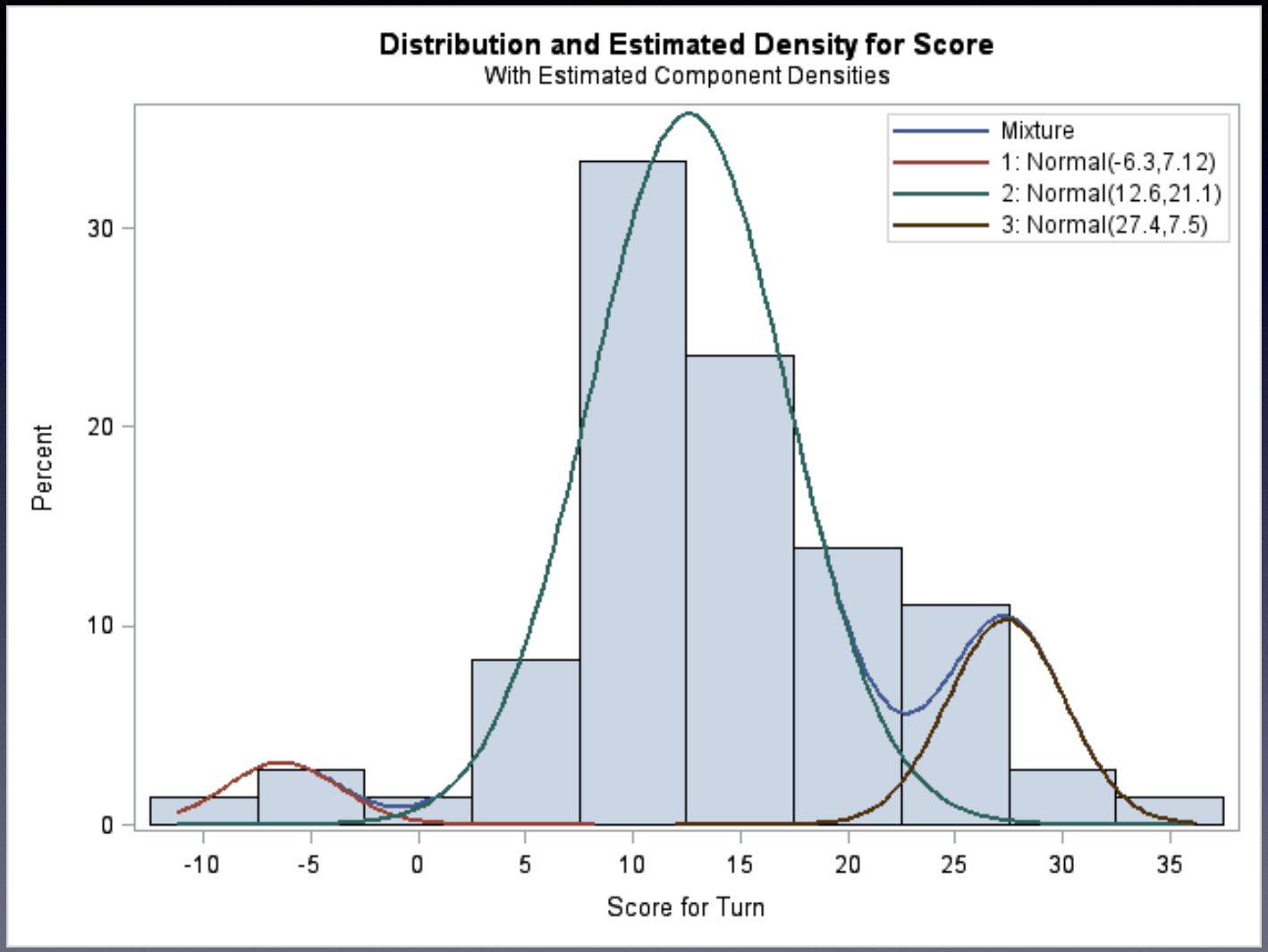
Density estimation is the act of estimating a continuous density field from a discretely sampled set of points drawn from that density field.

- Non-parametric Density Estimation (Tuesday)
  - Histograms (Knuth's, Bayesian Blocks)
  - Kernel Density Estimation (KDE)
  - (Bayesian) nearest neighbors method
- Parametric Density Estimation (today)
  - Gaussian Mixture Models
  - Extreme Deconvolution (XD)

Parametric Density Estimation

- KDE estimates the density of points by affixing a kernel to each point in the data set
- An alternative is to use fewer kernels than points, and fit for optimal location and width of each kernel
- This is known as a "**mixture model**"

Example (1-D)



http://blogs.sas.com/content/iml/2011/09/23/modeling-finite-mixtures-with-the-fmm-procedure.html

Mixture Models

- Notice the analogy:
  - KDE ≅ Interpolation
  - Mixture model ≅ Approximation

Mixture Models

- Mixture models can be interpreted in two ways:
  - The individual kernels are physically meaningless, and only the summed up approximation is interesting.
  - The locations and sizes of individual kernels reflect some underlying property of data; i.e., identify *clusters* or *classes* in the data.

Gaussian Mixture Model

- The most common mixture model uses Gaussian components, and is called a **Gaussian Mixture Model (GMM)**
- A GMM models the underlying density of points (the PDF) as a sum of Gaussians

The likelihood of a datum x_i for a Gaussian mixture model is given by

$$p(x_i|\boldsymbol{\theta}) = \sum_{j=1}^{M} \alpha_j \mathcal{N}(\mu_j, \sigma_j), \tag{4.18}$$

Gaussian Mixture Models

The likelihood of a datum x_i for a Gaussian mixture model is given by

$$p(x_i|\boldsymbol{\theta}) = \sum_{j=1}^{M} \alpha_j \mathcal{N}(\mu_j, \sigma_j), \tag{4.18}$$

where dependence on x_i comes via a Gaussian $\mathcal{N}(\mu_j, \sigma_j)$. The vector of parameters $\boldsymbol{\theta}$ that need to be estimated for a given data set $\{x_i\}$ includes normalization factors for each Gaussian, α_j, and its parameters μ_j and σ_j. It is assumed that the data have negligible uncertainties (e.g., compared

This is a model of the underlying density distribution. The parameters we need to estimate are:

- The locations of each gaussian, μ_j
- The widths of each gaussian, σ_j
- The amplitude of each gaussian, α_j

- The number of gaussian components, M

Gaussian Mixture Models

The likelihood of a datum x_i for a Gaussian mixture model is given by

$$p(x_i|\boldsymbol{\theta}) = \sum_{j=1}^{M} \alpha_j \mathcal{N}(\mu_j, \sigma_j), \tag{4.18}$$

where dependence on x_i comes via a Gaussian $\mathcal{N}(\mu_j, \sigma_j)$. The vector of parameters $\boldsymbol{\theta}$ that need to be estimated for a given data set $\{x_i\}$ includes normalization factors for each Gaussian, α_j, and its parameters μ_j and σ_j. It is assumed that the data have negligible uncertainties (e.g., compared

Usually solved using **Expectation Maximization algorithm**
(for a good tutorial see arXiv:statistics/1105.1476)

Expectation Maximization Algorithm

- **A well-known fast and straighforward solution developed by Dempster, Laird & Rubin (1977)**
- **Relies on the concept of Hidden Variables (classes)**

The likelihood of a datum x_i for a Gaussian mixture model is given by

$$p(x_i|\boldsymbol{\theta}) = \sum_{j=1}^{M} \alpha_j \mathcal{N}(\mu_j, \sigma_j),$$

There are (3M-1) parameters to estimate.

Since the class labels are not known, for each data value we can only determine the probability that it was generated by class j (sometimes called responsibility, e.g., HTF09). Given x_i, this probability can be obtained for each class using Bayes' rule (see eq. 3.10),

shorthand $w_{ij} = p(j|x_i)$

$$p(j|x_i) = \frac{\alpha_j \mathcal{N}(\mu_j, \sigma_j)}{\sum_{j=1}^{M} \alpha_j \mathcal{N}(\mu_j, \sigma_j)}. \qquad (4.21)$$

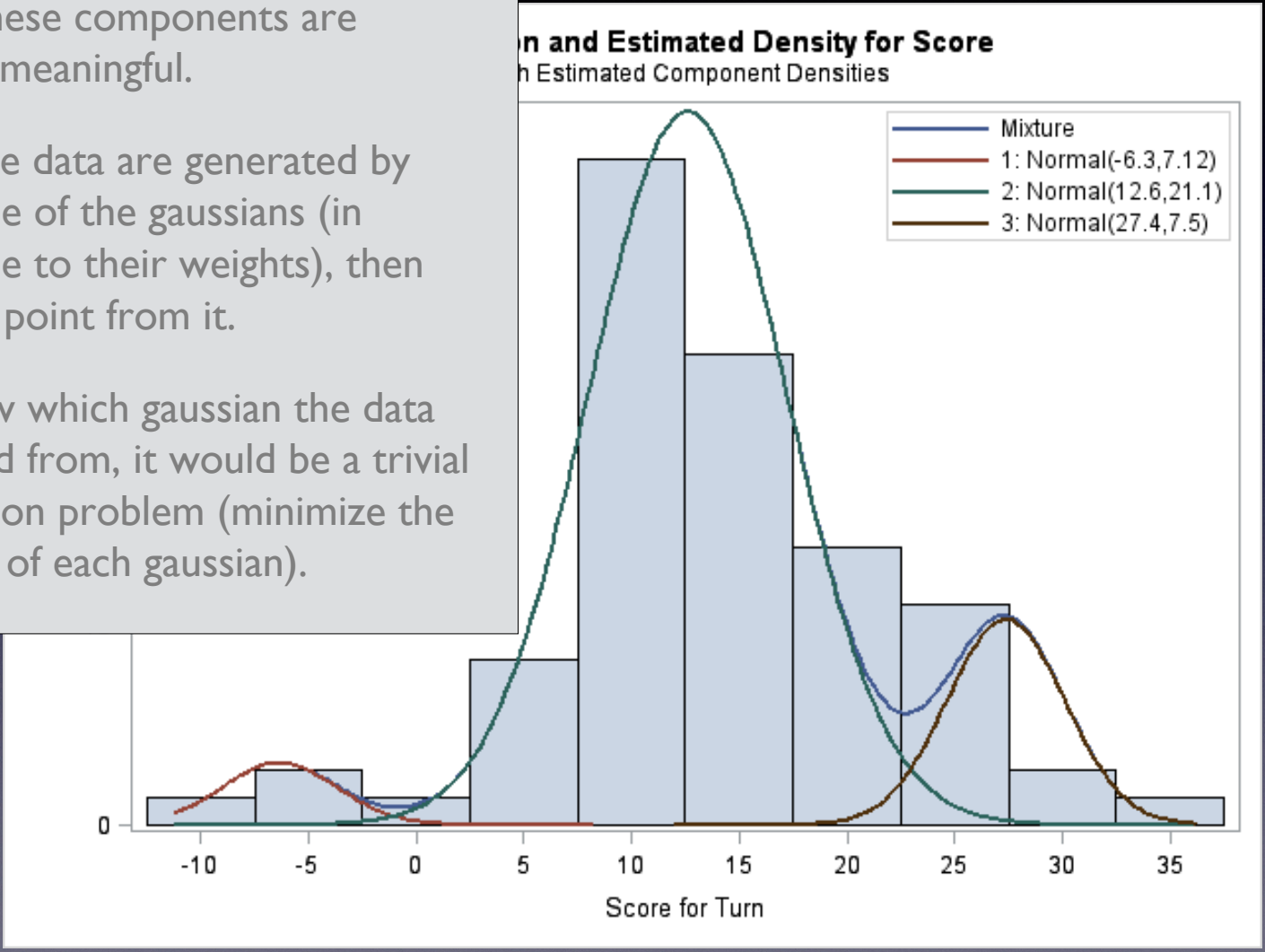
The class probability $p(j|x_i)$ is small when x_i is not within "a few" σ_j from μ_j (assuming that x_i is close to some other mixture component). Of course, $\sum_{j=1}^{M} p(j|x_i) = 1$. This probabilistic

Example (1-D)

Imagine these components are physically meaningful.

That is, the data are generated by picking one of the gaussians (in accordance to their weights), then drawing a point from it.

If we knew which gaussian the data was picked from, it would be a trivial minimization problem (minimize the likelihood of each gaussian).



http://blogs.sas.com/content/iml/2011/09/23/modeling-finite-mixtures-with-the-fmm-procedure.html

Expectation Maximization Algorithm

An iterative two-step (E and M) algorithm:

The key ingredient of the iterative EM algorithm is the assumption that the class probability $p(j|x_i)$ is known and fixed in each iteration (for a justification based on conditional probabilities, see [20] and HTF09). The EM algorithm is not limited to Gaussian mixtures, so instead of $\mathcal{N}(\mu_j, \sigma_j)$ in eq. 4.18, let us use a more general pdf for each component, $p_j(x_i|\boldsymbol{\theta})$ (for notational simplicity, we do not explicitly account for the fact that p_j includes only a subset of all $\boldsymbol{\theta}$ parameters, e.g., only μ_j and σ_j are relevant for the jth Gaussian component). By analogy with eq. 4.20, the log-likelihood is

$$\ln L = \sum_{i=1}^{N} \ln \left[\sum_{j=1}^{M} \alpha_j \, p_j(x_i|\boldsymbol{\theta}) \right] . \tag{4.22}$$

We can take a partial derivative of lnL with respect to the parameter θ_j,

$$\frac{\partial \ln L}{\partial \theta_j} = \sum_{i=1}^{N} \frac{\alpha_j}{\sum_{j=1}^{M} \alpha_j \, p_j(x_i|\boldsymbol{\theta})} \left[\frac{\partial p_j(x_i|\boldsymbol{\theta})}{\partial \theta_j} \right] \tag{4.23}$$

Expectation Maximization Algorithm

An iterative two-step (E and M) algorithm:

$$\frac{\partial \ln L}{\partial \theta_j} = -\sum_{i=1}^{N} w_{ij} \frac{\partial}{\partial \theta_j}\left[\ln \sigma_j + \frac{(x_i - \mu_j)^2}{2\sigma_j^2}\right], \tag{4.25}$$

where θ_j now corresponds to μ_j or σ_j. By setting the derivatives of lnL with respect to μ_j and σ_j to zero, we get the estimators (this derivation is discussed in more detail in §5.6.1)

$$\mu_j = \frac{\sum_{i=1}^{N} w_{ij} x_i}{\sum_{i=1}^{N} w_{ij}}, \tag{4.26}$$

$$\sigma_j^2 = \frac{\sum_{i=1}^{N} w_{ij} (x_i - \mu_j)^2}{\sum_{i=1}^{N} w_{ij}}, \tag{4.27}$$

and from the normalization constraint,

shorthand $w_{ij} = p(j|x_i)$

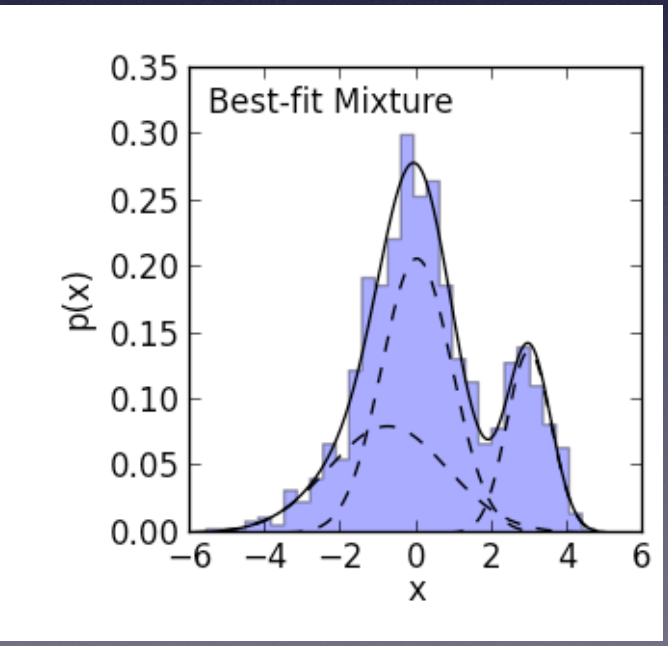
$$\alpha_j = \frac{1}{N} \sum_{i=1}^{N} w_{ij}. \tag{4.28}$$

These expressions and eq. 4.21 form the basis of the iterative EM algorithm in the case of Gaussian mixtures. Starting with a guess for w_{ij}, the values of α_j, μ_j, and σ_j are estimated using eqs. 4.26–4.28. This is the "maximization" *M-step* which brings the parameters closer toward the local maximum. In the subsequent "expectation" *E-step*, w_{ij} are updated using eq. 4.21. The algorithm is not sensitive to the initial guess of parameter values. For example, setting all σ_j to the sample standard deviation, all α_j to $1/M$, and randomly drawing μ_j from the observed $\{x_i\}$ values, typically works well in practice (see HTF09).

Expectation Maximization Algorithm

Scikit-learn contains an EM algorithm for fitting N-dimensional mixtures of Gaussians:

```
>>> import numpy as np
>>> from sklearn.mixture import GMM
>>> X = np.random.normal(size=(100, 1))  # 100 points in 1 dim
>>> model = GMM(2)  # two components
>>> model.fit(X)
>>> model.means_  # the locations of the best-fit components
array([[-0.05786756],
       [ 0.69668864]])
```



Gaussian Mixture Models

The likelihood of a datum x_i for a Gaussian mixture model is given by

$$p(x_i|\boldsymbol{\theta}) = \sum_{j=1}^{M} \alpha_j \mathcal{N}(\mu_j, \sigma_j), \tag{4.18}$$

where dependence on x_i comes via a Gaussian $\mathcal{N}(\mu_j, \sigma_j)$. The vector of parameters $\boldsymbol{\theta}$ that need to be estimated for a given data set $\{x_i\}$ includes normalization factors for each Gaussian, α_j, and its parameters μ_j and σ_j. It is assumed that the data have negligible uncertainties (e.g., compared

This is a model of the underlying density distribution. The parameters we need to estimate are:

- The locations of each gaussian, μ_j
- The widths of each gaussian, σ_j
- The amplitude of each gaussian, α_j

- **The number of gaussian components, M**

Gaussian Mixture Models

How to choose the number of classes?

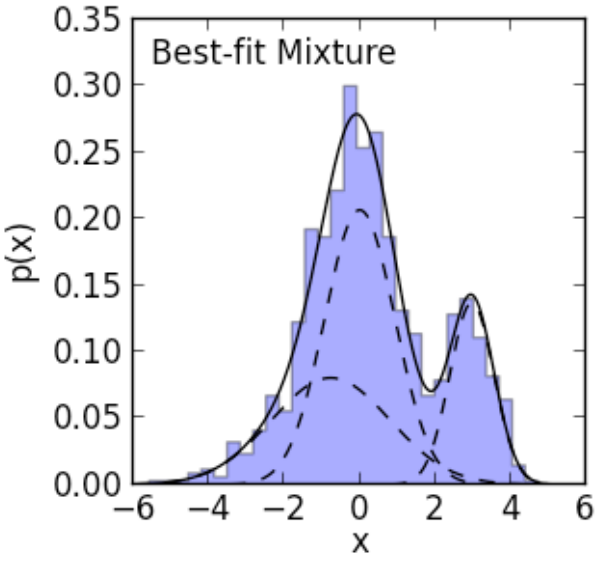
We have assumed in the above discussion of the EM algorithm that the number of classes in a mixture, M, is known. As M is increased, the description of the data set $\{x_i\}$ using a mixture model will steadily improve. On the other hand, a very large M is undesired—after all, $M = N$ will assign a mixture component to each point in a data set. How do we choose M in practice?

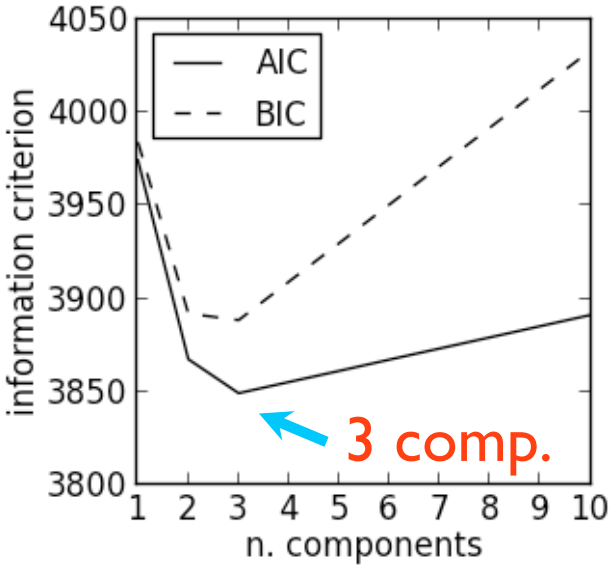
Usually determined using Bayesian Information Criterion (**BIC**),
or **cross-validation**

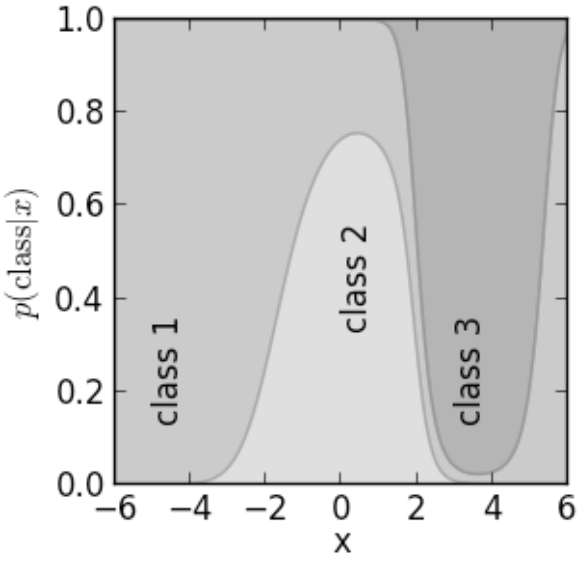
Gaussian Mixture Models

- make this plot by running
  %run fig_GMM_1D.py

This best GMM fit is also density estimation! The only difference compared to histograms is in the chosen fitting model function!







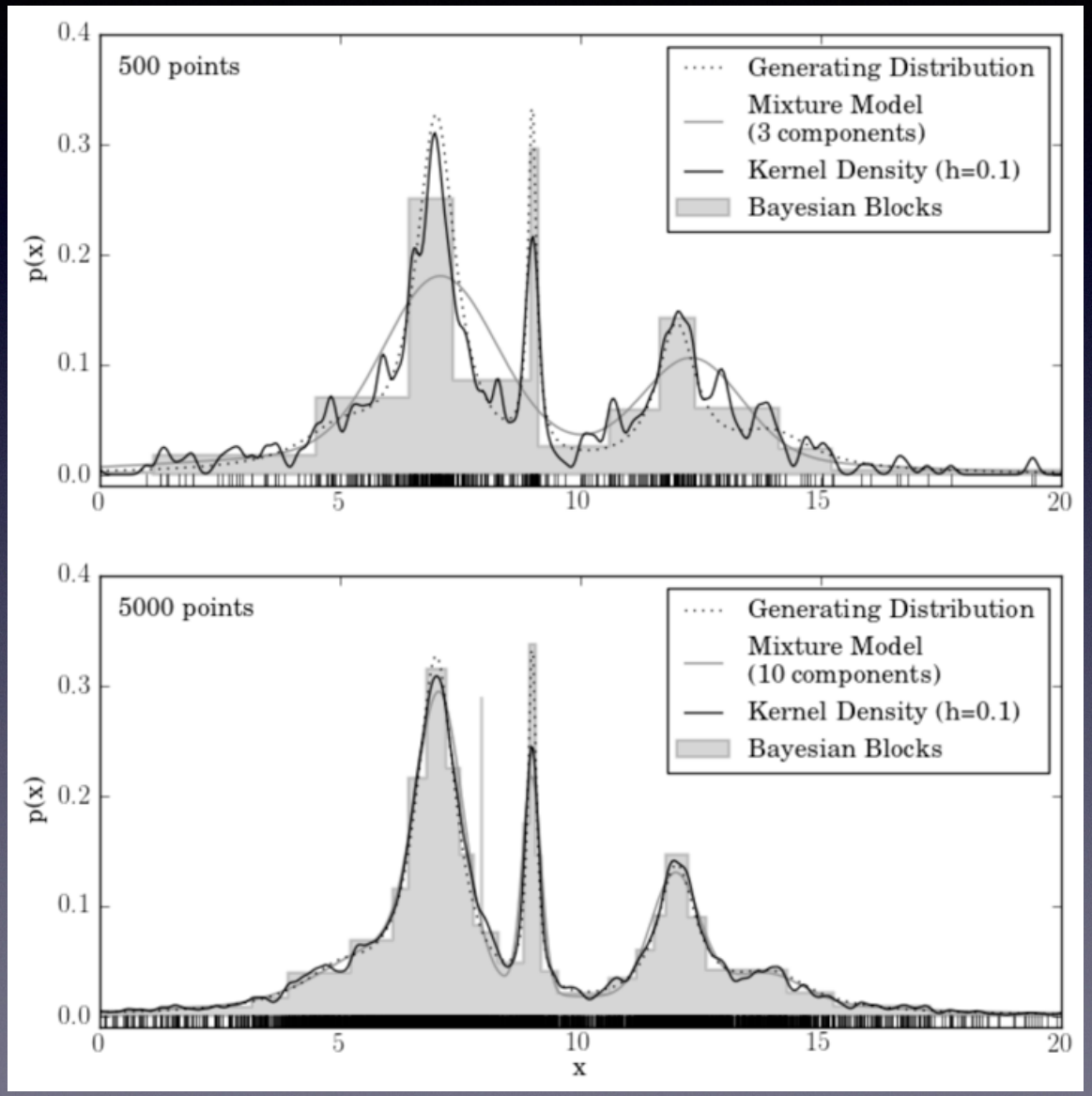
Uses Expectation Maximization method implemented in scikit-learn

Bayesian Information Criterion: it tells you how many components data support!

Example of classification (more next week)

Comparing different methods:

- you **could** make this plot by running
  %run fig_GMM_density_estimation.py



Key point: for large datasets, all methods result in similar estimates

puzzle: note the spike at x~8 for the BB algorithm in the bottom panel

Example in 2D

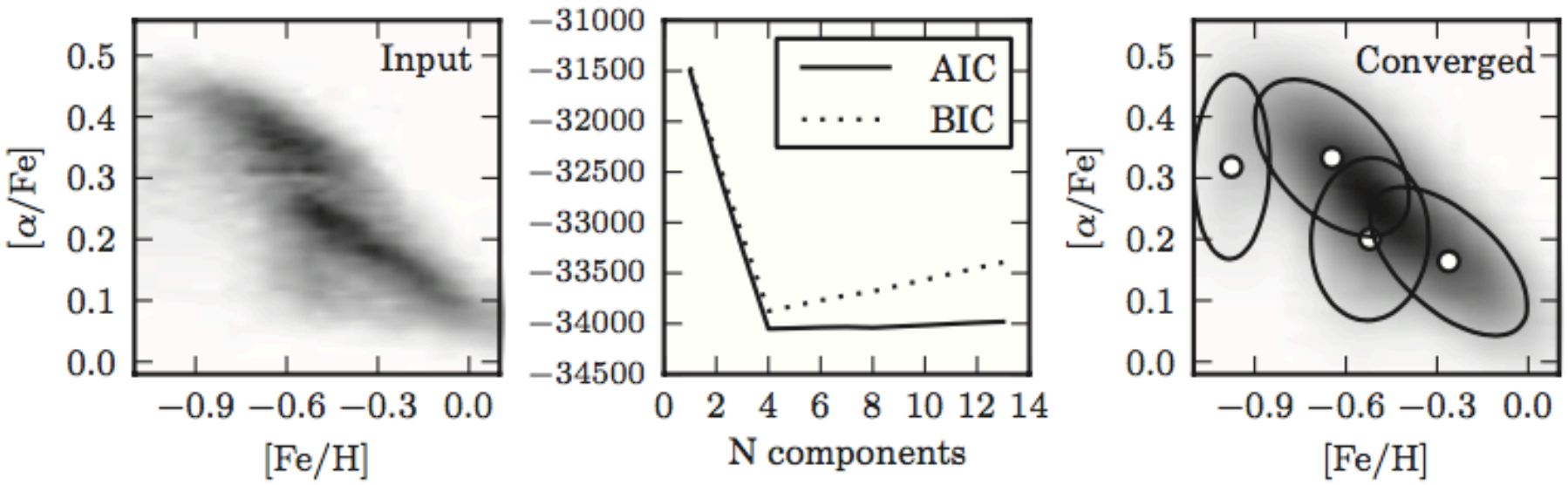


Figure 6.6. A two-dimensional mixture of Gaussians for the stellar metallicity data. The left panel shows the number density of stars as a function of two measures of their chemical composition: metallicity ([Fe/H]) and α-element abundance ([α/Fe]). The right panel shows the density estimated using mixtures of Gaussians together with the positions and covariances (2σ levels) of those Gaussians. The center panel compares the information criteria AIC and BIC (see §4.3.2 and §5.4.3).

- Note: The fact that the information criteria, such as BIC/AIC, prefer an *N*-component peak does not necessarily mean that there are *N* components.
- If the clusters in the input data are not near Gaussian, or if there is a strong background, the number of Gaussian components in the mixture will not generally correspond to the number of clusters in the data.

Example in 2D

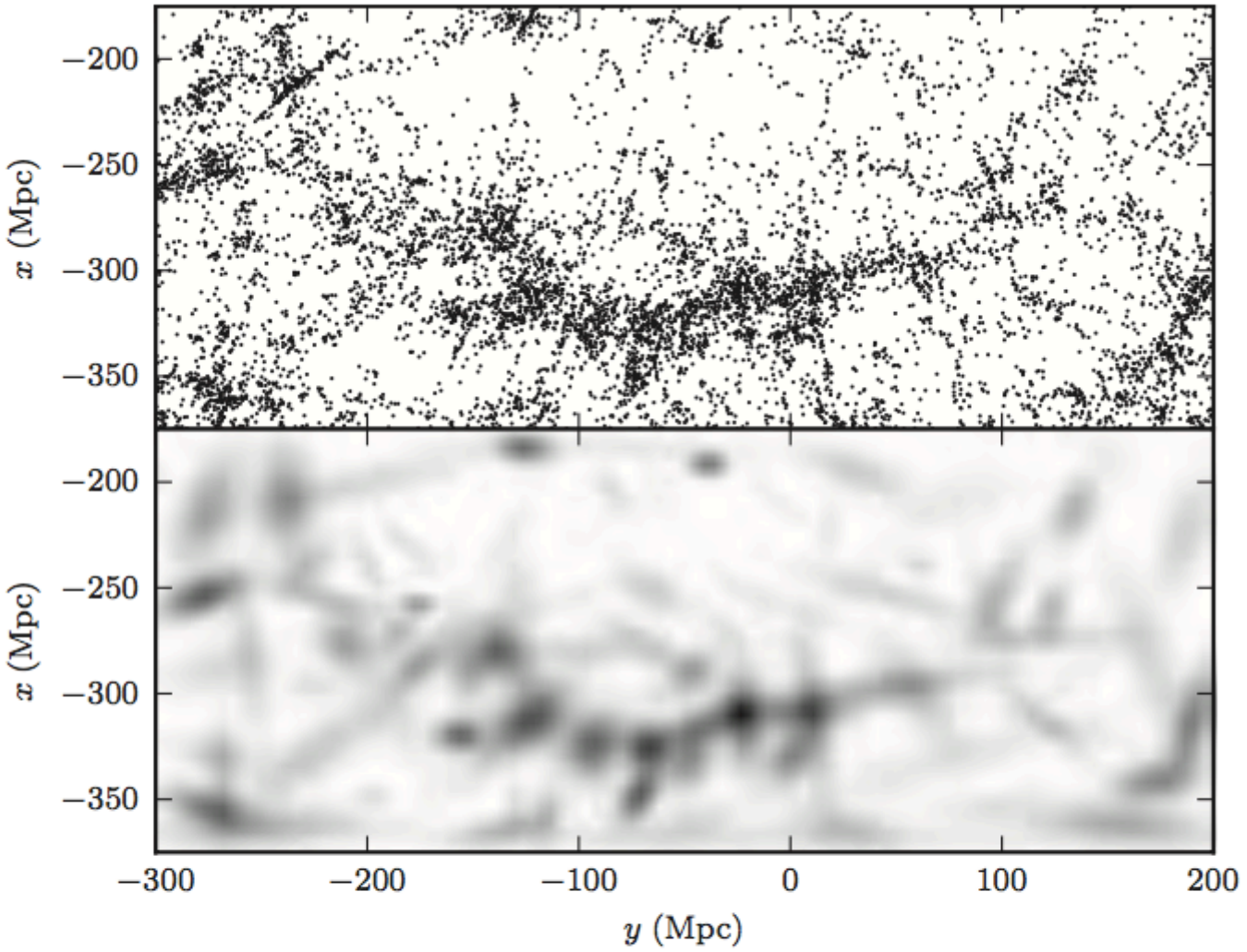
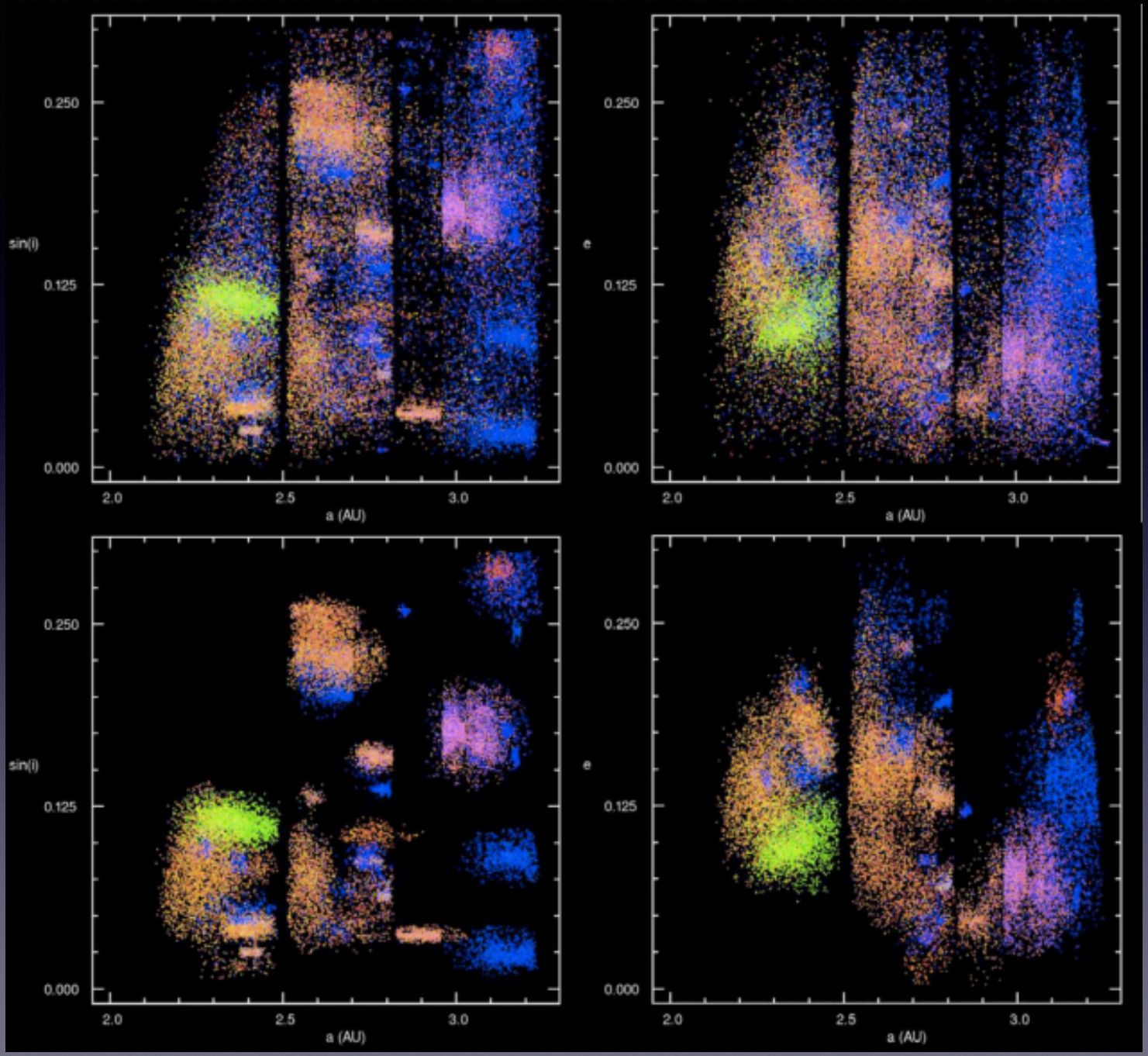


Figure 6.7. A two-dimensional mixture of 100 Gaussians (bottom) used to estimate the number density distribution of galaxies within the SDSS Great Wall (top). Compare to figures 6.3 and 6.4, where the density for the same distribution is computed using both kernel density and nearest-neighbor-based estimates.

Solar System Asteroids in Color and "Orbital Elements Space"

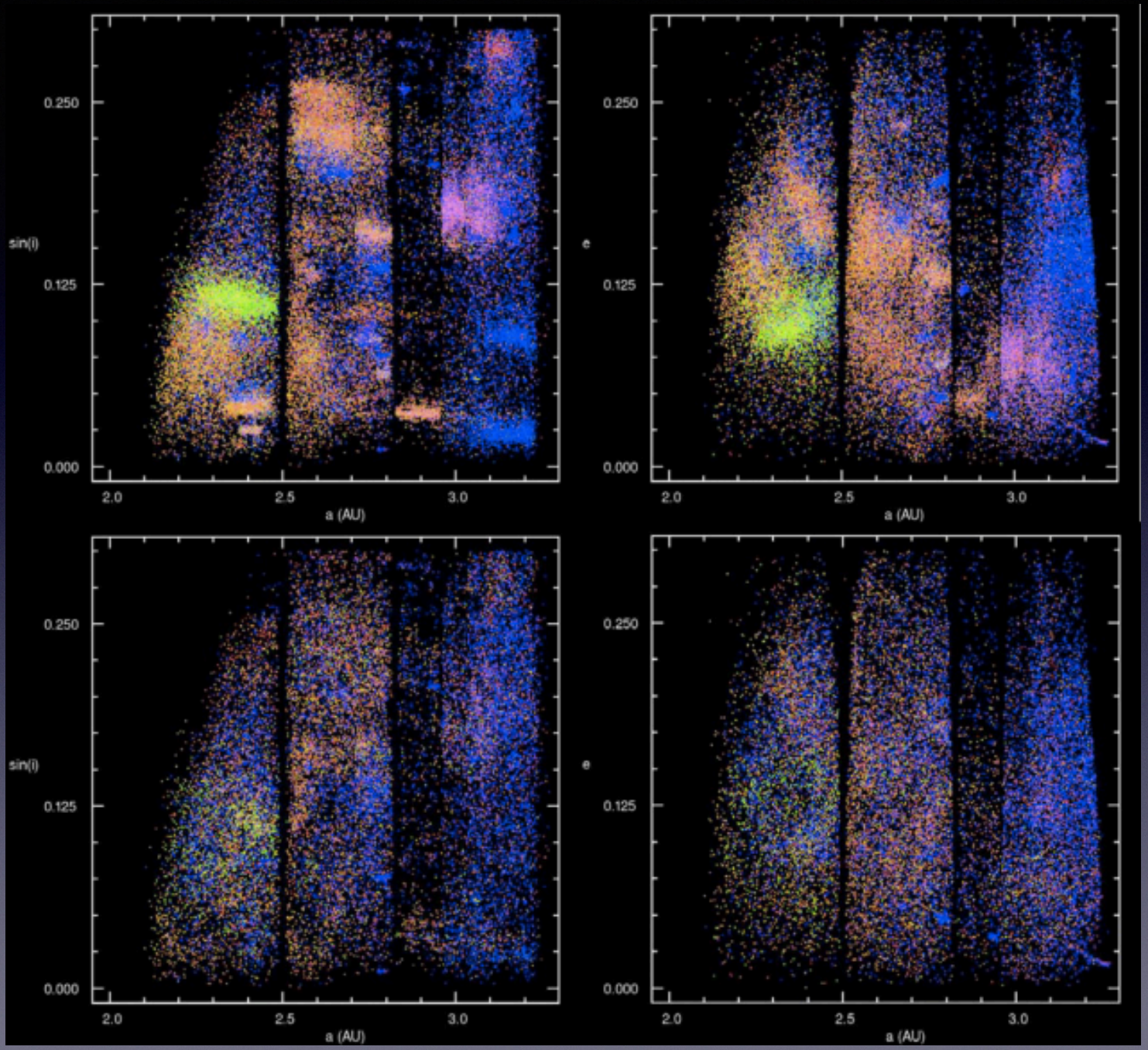
Notice the asteroid families!



Parker et al. 2008

Solar System Asteroids in Color and "Orbital Elements Space"

Notice the asteroid families!



Parker et al. 2008

Gaussian Mixture Models with Errors

The previous example assumed that uncertainty in data values (x) was negligible. What do we do when this is not the case?

For example, in case of homescedastic Gaussian measurement errors, the three GMM components from the last example would be broadened (the measurement error would be added in quadrature to their intrinsic widths).

A recently introduced application of GMM with errors in astronomical context was dubbed **"Extreme Deconvolution"** (see Bovy et al. 2009, ArXiv:0905.2979)

Extreme Deconvolution works in multi-dimensional spaces too, and naturally treats noisy, heterogeneous, and incomplete data.

More details and a high-D example later...

Extreme Deconvolution in high-D

- … But let's first warm up with a toy 1-D case
- Two basic assumptions:
  a) the sampled population distribution is a single Gaussian, and
  b) data have known heteroscedastic errors:

In order to proceed with parameter estimation in this situation, we shall assume that data were drawn from an intrinsic $\mathcal{N}(\mu, \sigma)$ distribution, and that measurement errors are also Gaussian and described by the known width e_i. Starting with an analog of eq. 5.52,

$$p(\{x_i\}|\mu, \sigma, I) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi}(\sigma^2 + e_i^2)^{1/2}} \exp\left(\frac{-(x_i - \mu)^2}{2(\sigma^2 + e_i^2)}\right), \quad (5.63)$$

and following the same steps as above (with uniform priors for μ and σ), we get an analog of eq. 5.56:

$$L_p = \text{constant} - \frac{1}{2}\sum_{i=1}^{N}\left(\ln(\sigma^2 + e_i^2) + \frac{(x_i - \mu)^2}{\sigma^2 + e_i^2}\right). \quad (5.64)$$

The solution for best-fit parameters cannot be expressed in a closed form any more (it can for homoscedastic errors)

- **We can evaluate posterior pdf by brute force:**

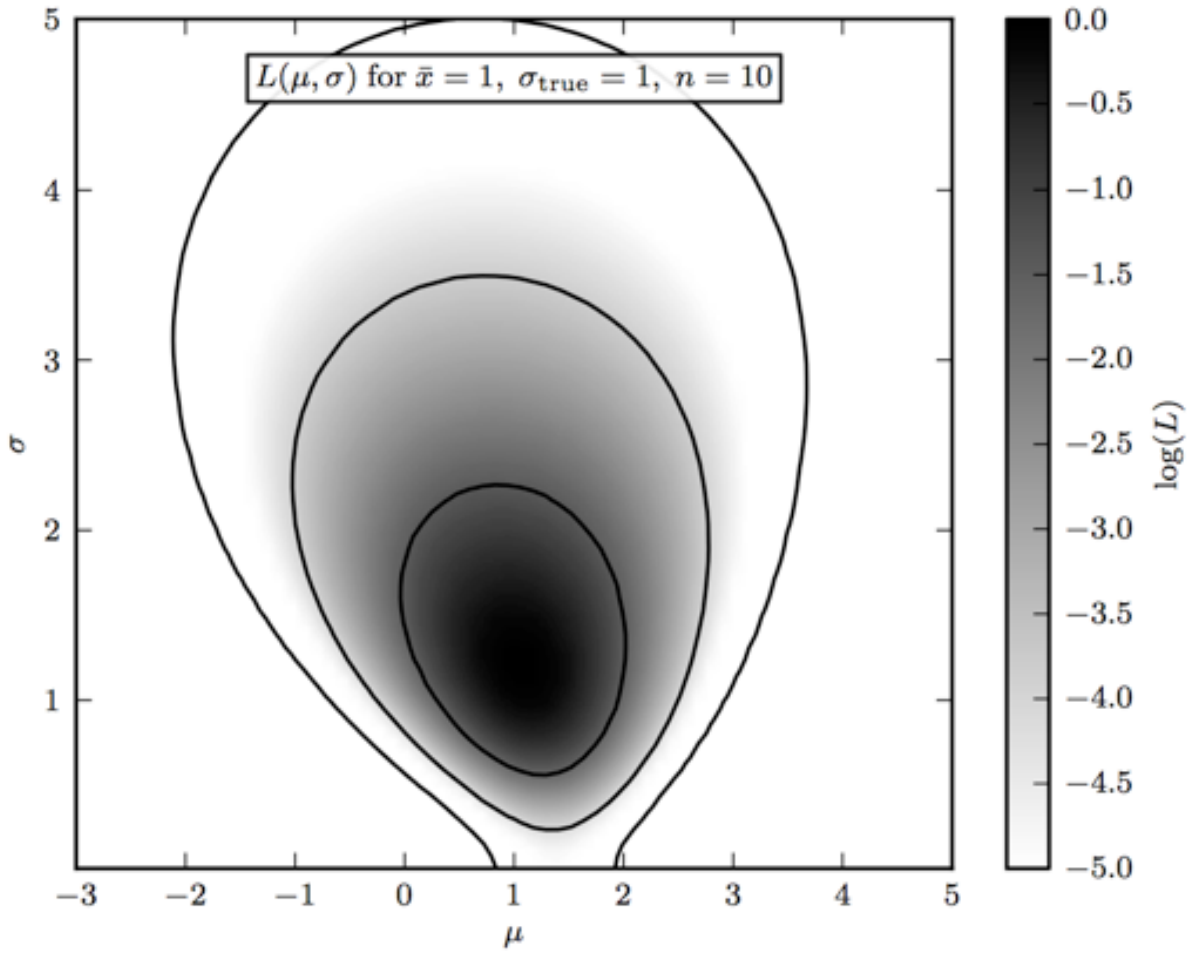


Figure 5.7.: The logarithm of the posterior probability density function for μ and σ, $L_p(\mu, \sigma)$, for a Gaussian distribution with heteroscedastic Gaussian measurement errors (sampled uniformly from the 0–3 interval), given by eq. 5.64. The input values are $\mu = 1$ and $\sigma = 1$, and a randomly generated sample has 10 points. Note that the posterior pdf is not symmetric with respect to the $\mu = 1$ line, and that the outermost contour, which encloses the region that contains 0.997 of the cumulative (integrated) posterior probability, allows solutions with $\sigma = 0$.

A more complicated (realistic) model:

- **Two basic assumptions:**
  a) the sampled population distribution is a arbitrary mixture of high-D Gaussian components, and
  b) data have heteroscedastic errors with known covariance matrix

being α_i. Thus, the pdf of $\mathbf{x}$ is given as

$$p(\mathbf{x}) = \sum_j \alpha_j \mathcal{N}(\mathbf{x}|\mu_j, \mathbf{\Sigma}_j)\,, \tag{6.18}$$

where, recalling eq. 3.97,

$$\mathcal{N}(\mathbf{x}|\mu_j, \mathbf{\Sigma}_j) = \frac{1}{\sqrt{(2\pi)^D \det(\mathbf{\Sigma}_j)}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^T \mathbf{\Sigma}_j^{-1}(\mathbf{x}-\mu)\right). \tag{6.19}$$

Extreme deconvolution generalizes the EM approach to a case with measurement errors. More explicitly, one assumes that the noisy observations $\mathbf{x}_i$ and the true values $\mathbf{v}_i$ are related through

$$\mathbf{x}_i = \mathbf{R}_i \mathbf{v}_i + \epsilon_i, \tag{6.20}$$

If we can minimize these parameters, we have a very powerful method (and it can treat incomplete and heterogeneous data, too)!

Extreme Deconvolution in high-D (XD)

Bovy, Hogg & Roweis 2011 (arXiv:0905.2979)

- The expectation (E) step:

$$
\begin{aligned}
q_{ij} &\leftarrow \frac{\alpha_j \mathcal{N}(\mathbf{w}_i|\mathbf{R}_i\mu_j, \mathbf{T}_{ij})}{\sum_j \alpha_k \mathcal{N}(\mathbf{w}_i|\mathbf{R}_i\mu_k, \mathbf{T}_{ij})}, \\
\mathbf{b}_{ij} &\leftarrow \mu_j + \mathbf{\Sigma}_j \mathbf{R}_i^T \mathbf{T}_{ij}^{-1}(\mathbf{w}_i - \mathbf{R}_i\mu_j), \\
\mathbf{B}_{ij} &\leftarrow \mathbf{\Sigma}_j - \mathbf{\Sigma}_j \mathbf{R}_i^T \mathbf{T}_{ij}^{-1} \mathbf{R}_i \mathbf{\Sigma}_j,
\end{aligned}
$$

  where $\mathbf{T}_{ij} = \mathbf{R}_i \mathbf{\Sigma}_j \mathbf{R}_i^T + \mathbf{S}_i$.

- The maximization (M) step:

$$
\begin{aligned}
\alpha_i &\leftarrow \frac{1}{N} \sum_i q_{ij}, \\
\mu_j &\leftarrow \frac{1}{q_j} \sum_i q_{ij} \mathbf{b}_{ij}, \\
\mathbf{\Sigma}_j &\leftarrow \frac{1}{q_j} \sum_i q_{ij} [(\mu_j - \mathbf{b}_{ij})(\mu_j - \mathbf{b}_{ij}^T) + \mathbf{B}_{ij}],
\end{aligned}
$$

  where $q_j = \sum_i q_{ij}$.

Extreme Deconvolution in high-D (XD)

Bovy, Hogg & Roweis 2011 (arXiv:0905.2979)

The iteration of these steps increases the likelihood of the observations $\mathbf{w}_i$, given the model parameters. Thus, iterating until convergence, one obtains a solution that is a local maximum of the likelihood. This method has been used with success in quasar classification, by estimating the densities of quasar and nonquasar objects from flux measurements; see [5]. Details of the use of XD, including methods to avoid local maxima in the likelihood surface, can be found in [6].

AstroML contains an implementation of XD which has a similar interface to `GMM` in Scikit-learn:

```
import numpy as np
from astroML.density_estimation import XDGMM

X = np.random.normal(size=(1000, 1))  # 1000 pts in 1 dim
Xerr = np.random.random((1000, 1, 1))  # 1000 1x1 covariance matrices
xdgmm = XDGMM(n_components=2)
xdgmm.fit(X, Xerr)  # fit the model
logp = xdgmm.logprob_a(X, Xerr)  # evaluate probability
X_new = xdgmm.sample(1000) # sample new points from distribution
```

Extreme Deconvolution in high-D (XD)

- make this plot by running %run fig_XD_example.py

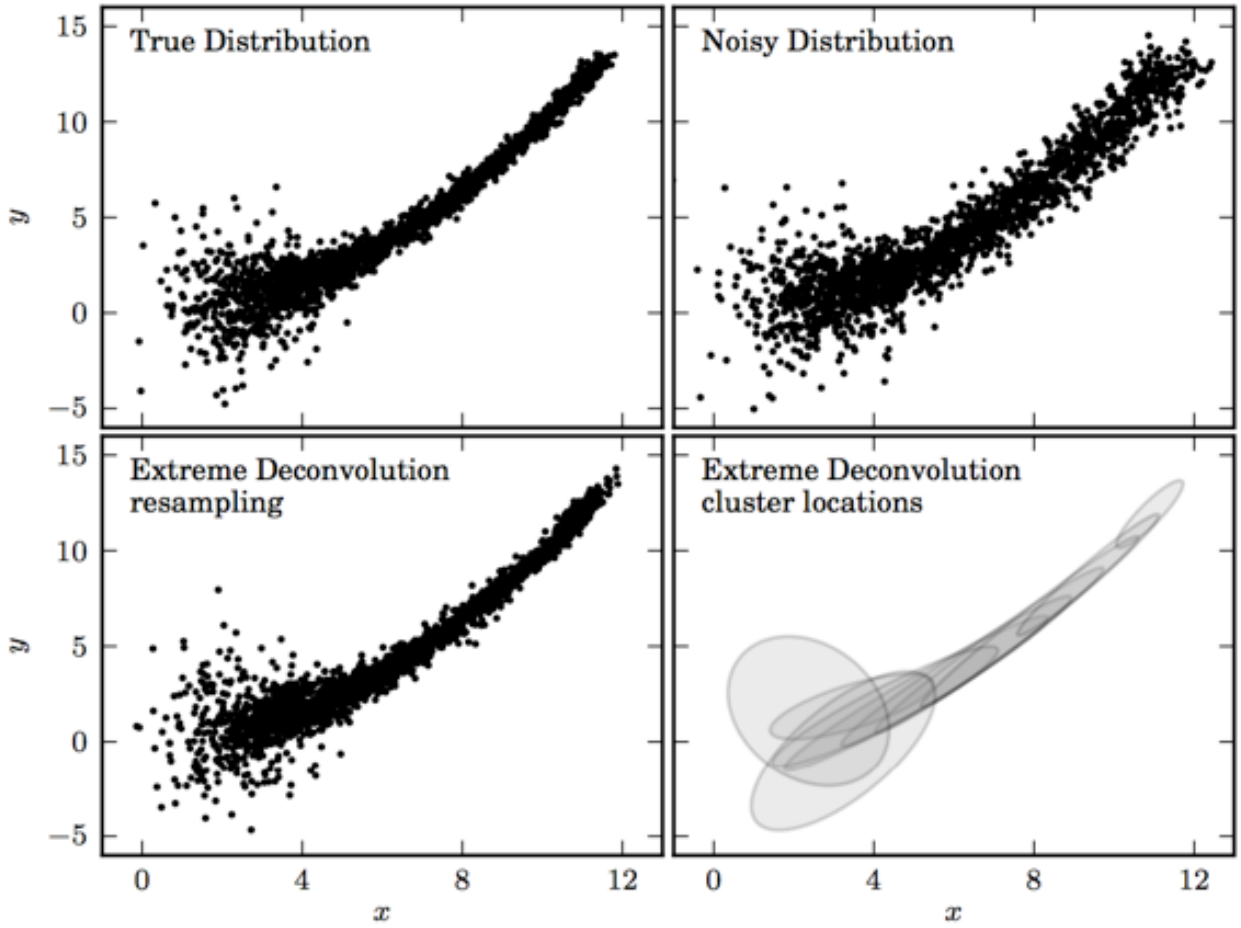
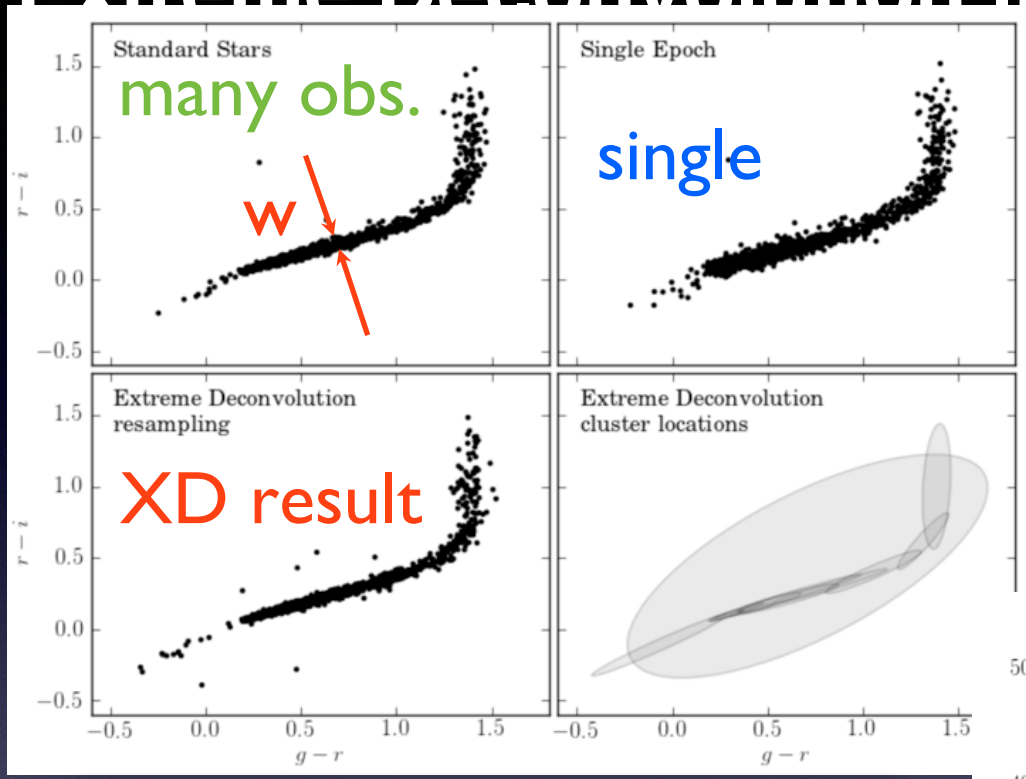
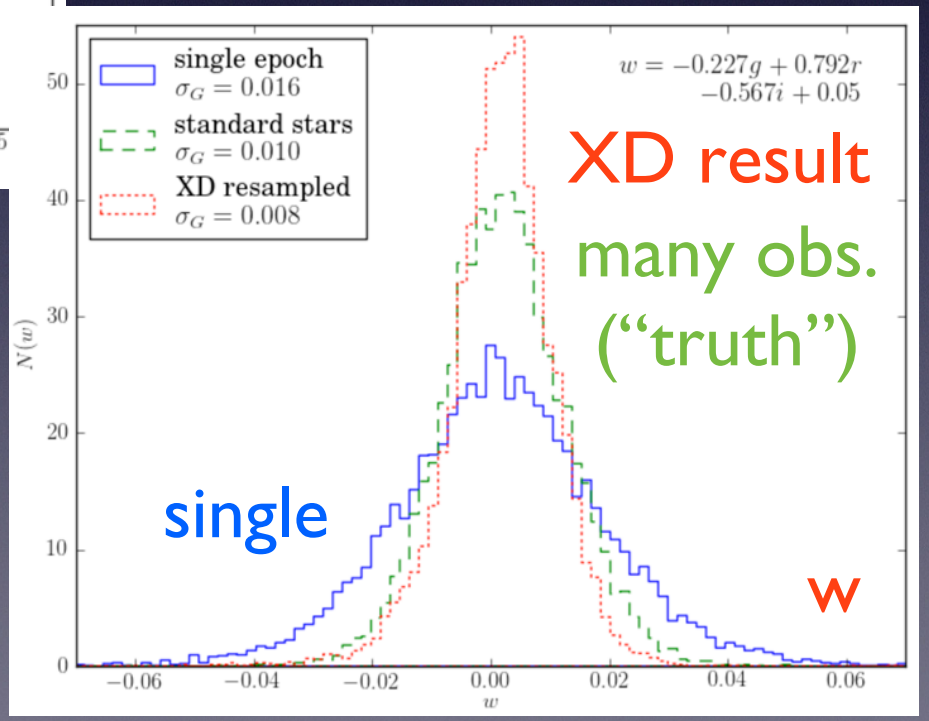


Figure 6.11.: An example of extreme deconvolution showing a simulated two-dimensional distribution of points, where the positions are subject to errors. The top two panels show the distributions with small (left) and large (right) errors. The bottom panels show the densities derived from the noisy sample (top-right panel) using extreme deconvolution; the resulting distribution closely matches that shown in the top-left panel.

Extreme Deconvolution in high-D (XD)



XD is a very powerful method: if GMM is able to model the observed distribution (usually OK)



XD result: the intrinsic width of the r-i vs. g-r stellar locus is ~0.01 mag., in agreement with high-SNR averaged multi-epoch photometry

Homework #9

- Take the results of queries from Homework 7, and run KDE on the (g-r, r) color-magnitude diagram