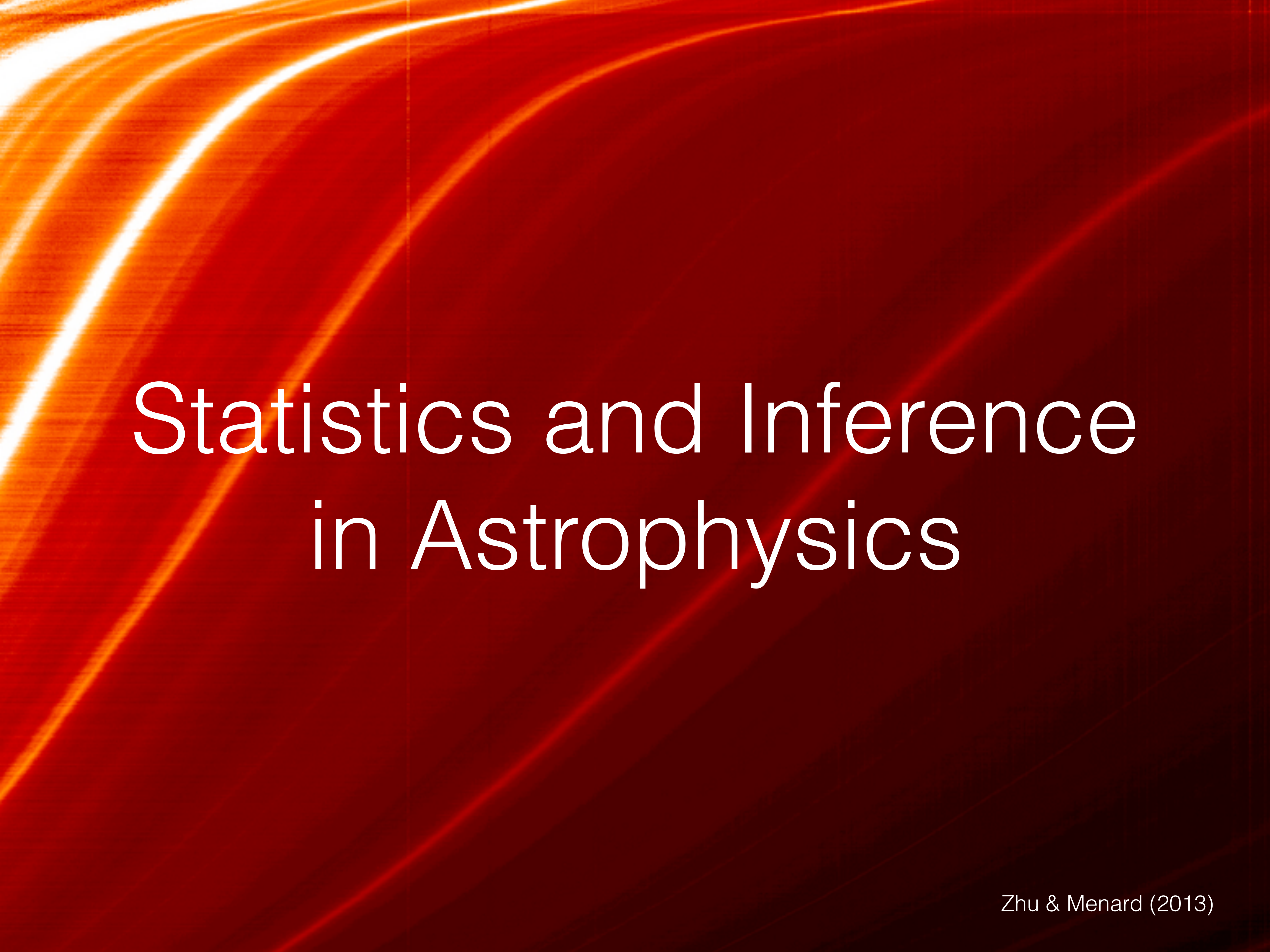
Statistics and Inference in Astrophysics



Zhu & Menard (2013)

Today: brief intro to machine learning tools

Machine learning

- Techniques to learn patterns in the data in flexible way; not parameter inference
- Main tasks:
    - Density estimation and clustering: What is the distribution of the data?
    - Dimensionality reduction: What are the most important dimensions in the data?
    - Regression: Learn to predict y(x) from (x,y) training data
    - Classification: Learn to predict classification labels L from (x,L) training data
- Distinction between *supervised* and *unsupervised* learning

Density estimation

- Have data points $\{x_i\}$ —> what is the density $\rho(x)$?

- Saw this in bootstrap: $\rho(x) = \Sigma_i \, \delta(x-x_i)$

- Parametric: fit $\rho(x)$ with some functional form with parameters θ, e.g., $\rho(x) = N(x|\text{mean},\text{variance})$ —> use parameter-inference techniques from L2

- Non-parametric: Similar to sum-of-delta functions, but replace delta function with a different function —> build $\rho(x)$ directly from the data without parameters

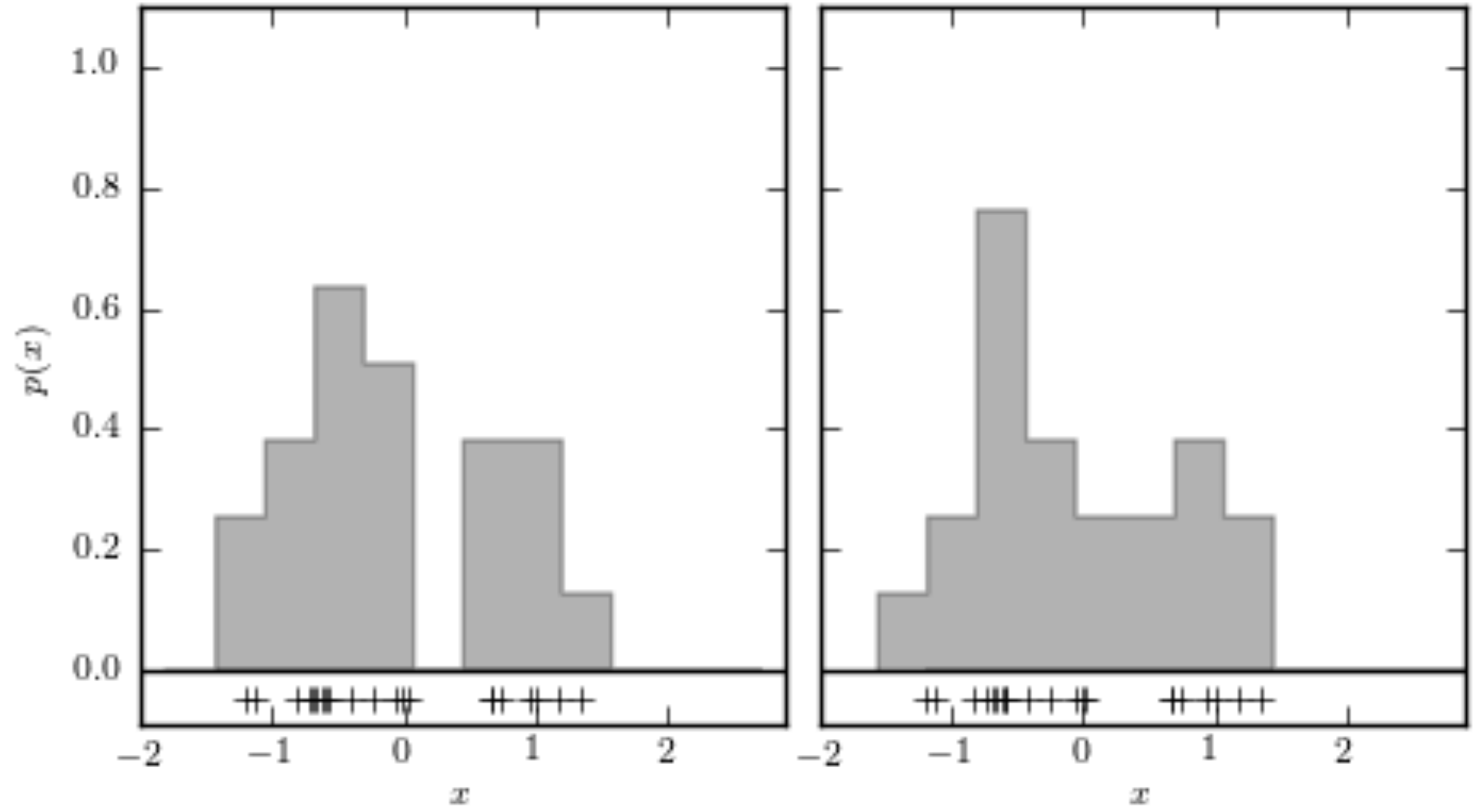
Simple parametric density estimation

- For example, model ρ(x) as Gaussian with mean *m* and variance *v*
- Data $\{x_i\}$, independently drawn w/o error
- Likelihood for individual x_i: $L_i = N(x_i|m,v)$
- Posterior PDF = $\mathrm{Prod}_i\ L_i$
- Optimizing this gives
  m = mean(x_i),
  v = (N-1)/N variance(x_i)
- No closed-form when data points have individual uncertainties σ_i

Simple non-parametric density estimation: histogram

- A histogram is a form of density estimation
- Non-parametric because histogram per se does not have explicit parameters
- But have *hyperparameters*: location and width of bins that need to be chosen; hyperparameters don’t directly set the density, but constrain, e.g., it smoothness
- Widely used, but often doesn’t give a good representation of the data, non-smooth, and difficult in higher dimensions

Histogram example



Same data, different binning!

Ivezic et al. (2014)

Kernel density estimation (KDE)

- Remember from bootstrap: $\rho(x) = \Sigma_i \delta(x-x_i)$
- Replace $\delta(.)$ with a

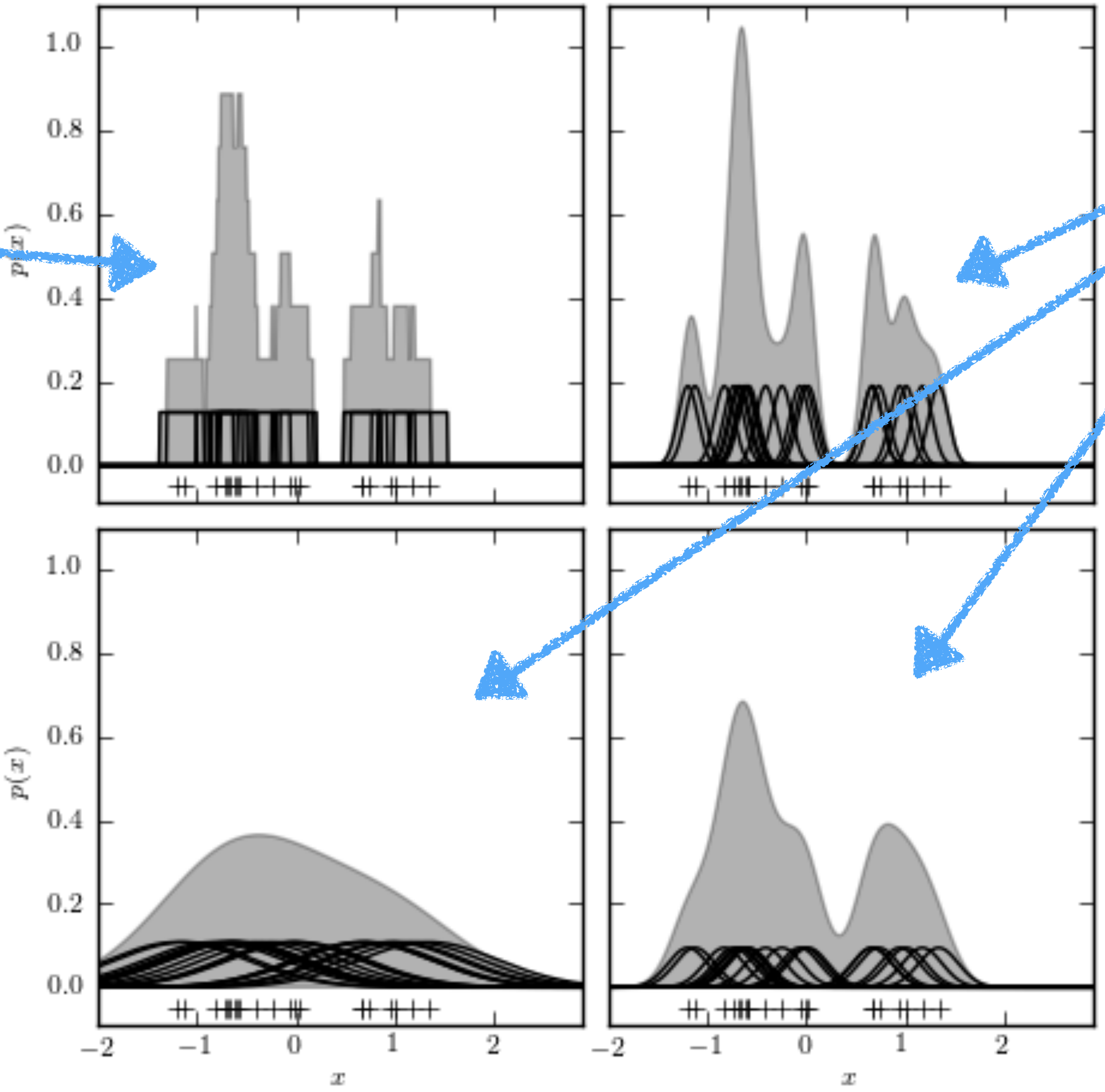
  kernel $K(.)$ with width h
  $x-x_i$ with distance function $d(x,x_i)$:

  $\rho(x) = \Sigma_i K(d[x,x_i]/h_i)$
- $K(.)$ could be: tophat function, similar to histogram, a Gaussian, or …

KDE example

Tophat

Gaussian with different widths



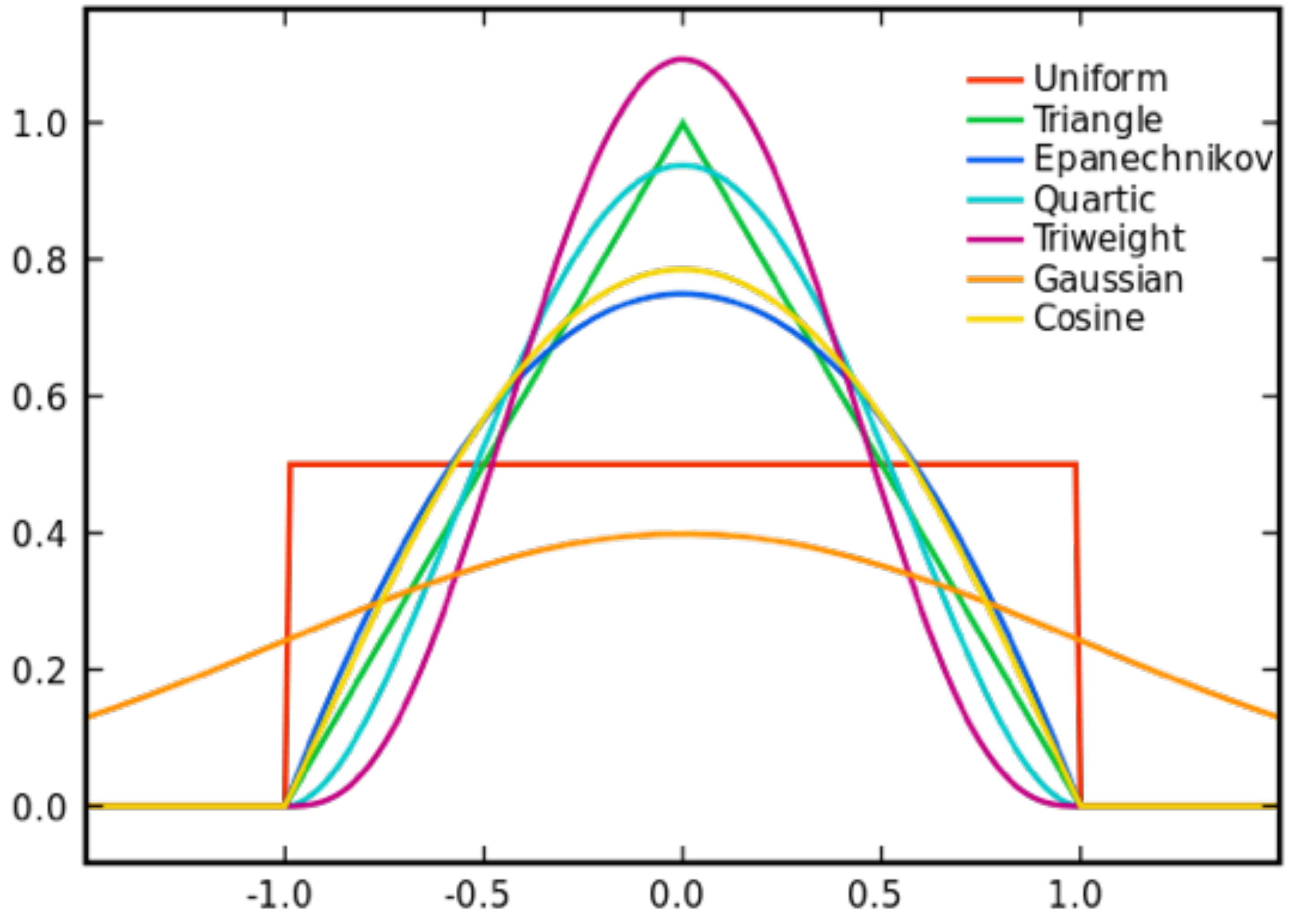
Ivezic et al. (2014)

KDE: kernels

- Kernels: symmetric functions around zero, positive everywhere, integrate to 1

- Gaussian convenient, but has infinite support: need to always use all points to get a density evaluation

- Epanechnikov optimal in that it gives the smallest expected mean-squared-error: $K(r = d(x,x_i)) = 3(1-r^2)/4$, $r <= 1$



Brian Amberg/Wikipedia

KDE: bandwidth

- Need to set width *h* of the kernel, this is a *hyperparameter*

- Some rules-of-thumb based on Gaussian data: Scott's rule: $h = N^{-1/(dim+4)}$ [if data scaled to have unit variance]
Silverman's rule: $h = [N*(dim+2)/4]^{-1/(dim+4)}$ [same scaling]

- Other way: leave-one-out-cross-validation (see last lecture)

- Or minimize Mean-Integrated-Square-Error

- Can also have variable h that depends on the local density:
$h(x) = k / [\rho(x)]^{1/dim}$,
higher density —> smaller kernel width

KDE applications

- Easy-to-use and standard tool when you need to estimate a density

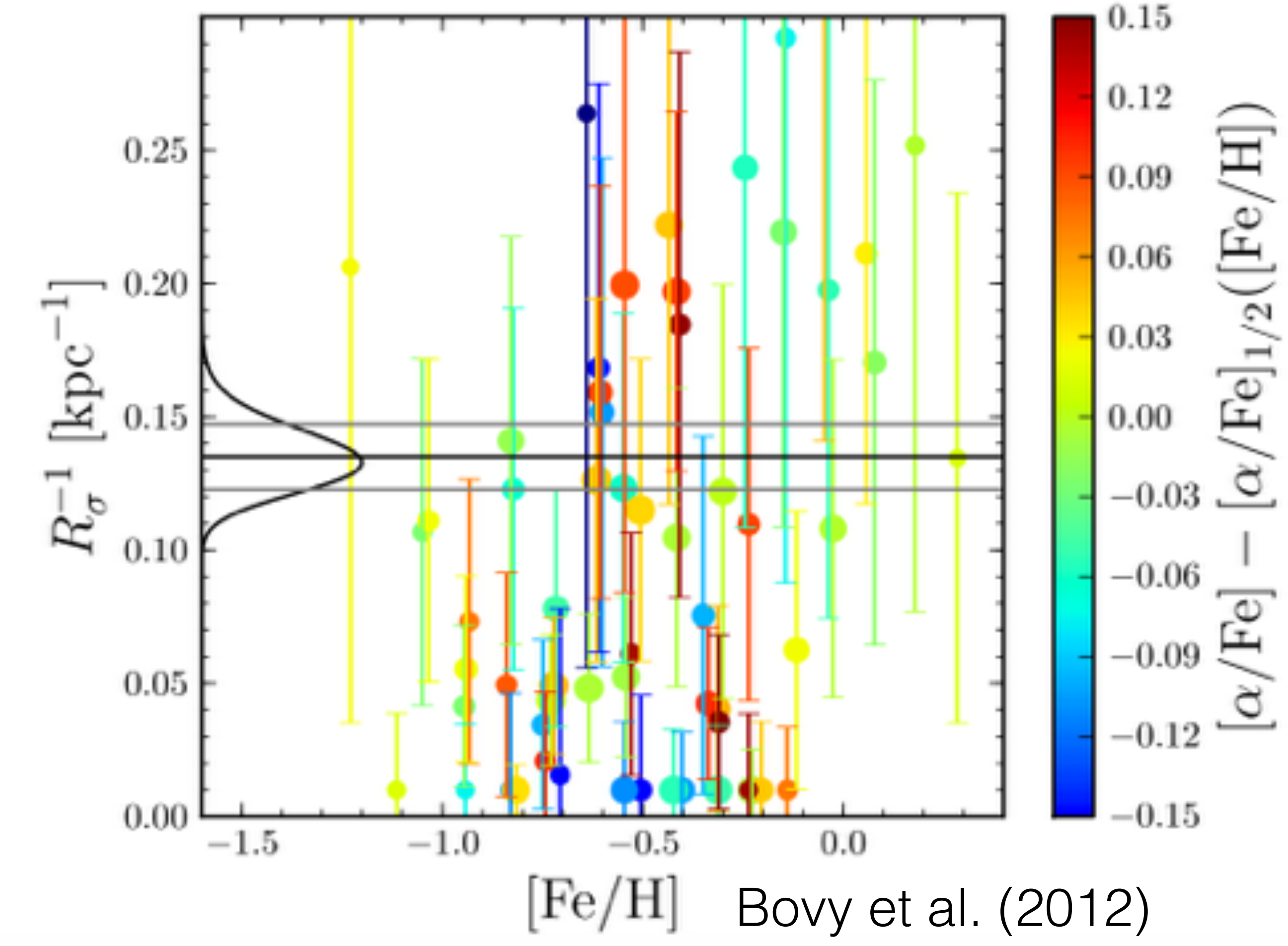
- Examples:

  - PDF from MCMC samples

  - You have run a bunch of simulations that give points in some space (e.g., stellar tracks with MESA) and want to estimate a density covering the whole space

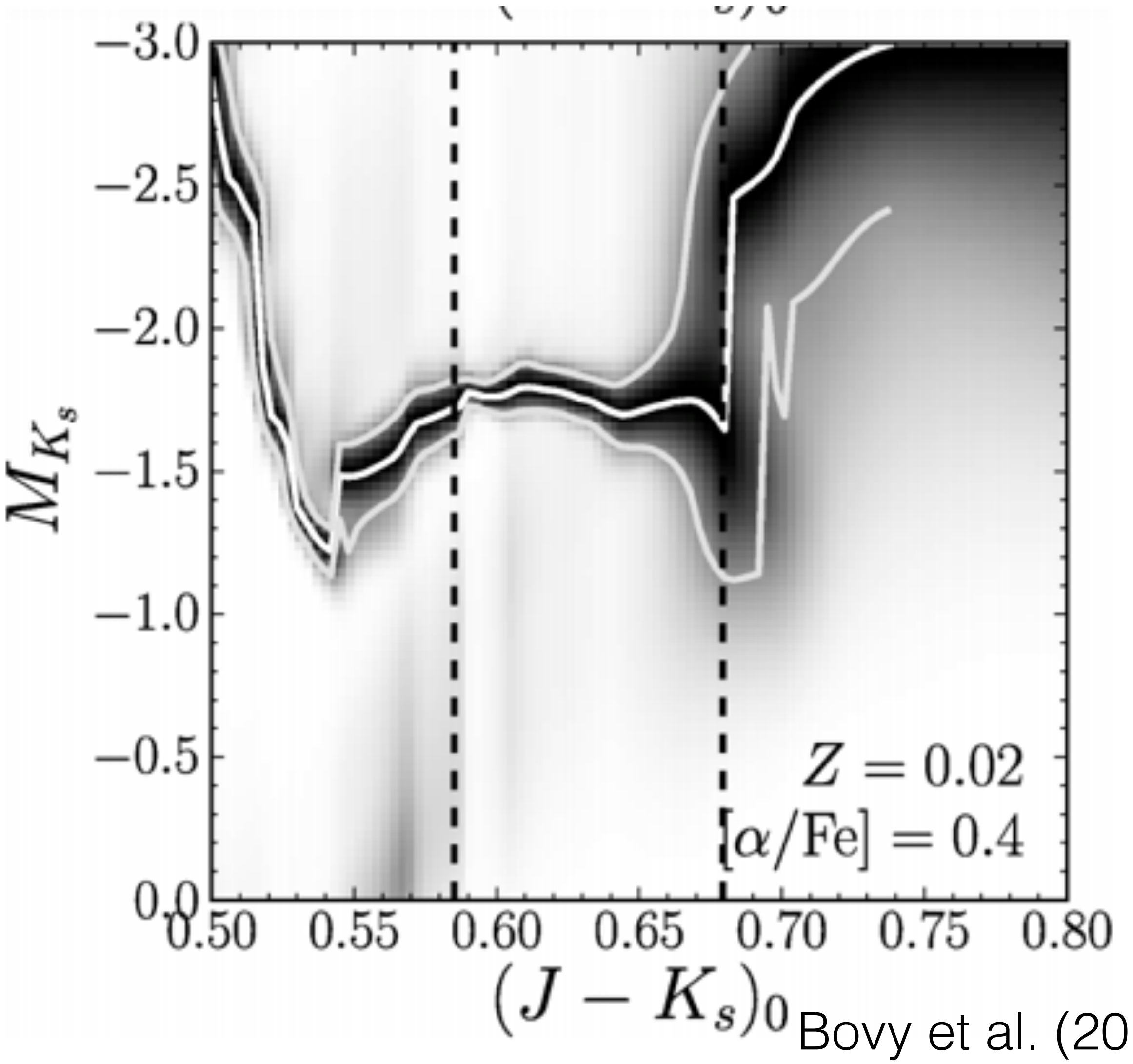
- But difficult to apply when data points have errors and want to deconvolve

Some examples…



Bovy et al. (2012)

MCMC chain —> KDE PDF



Bovy et al. (2014)

Theoretical model points —> KDE density

Parametric density estimation with many parameters: Gaussian mixtures

- Single Gaussian: strongly constrained parametric model; KDE w/ Gaussian kernel: very flexible, but as many components as data points

- Gaussian Mixture Model (GMM): in between: model density $\rho(x)$ as sum of K Gaussians, $K < N$

- Parameters: amplitudes, means, and variances of all Gaussians

- $\rho(x) = \Sigma_k\, a_k\, N(x|m_k,V_k)$

- Could optimize likelihood for all parameters….

GMM and EM

- When *K* becomes large, many parameters —> high-dimensional parameter space to search for optimal solution
- Expectation-Maximization algorithm: General algorithm to optimize these kinds of problems
- Add a q_{ik} assignment variable to each data point: data point i drawn from component k where $q_{ik} = 1$ (all other $q_{ik}= 0$)
- If we knew all q_i, then optimizing would be easy:

  $a_k = 1/N \Sigma_i q_{ik}$
  mean_k = mean of those x_i with $q_{ik} = 1$
  variance_k = variance of those x_i with $q_{ik} = 1$

GMM and EM

- Expectation-maximization: Can show that following two steps always increase likelihood

  E(xpectation):
  q_{ik} = a_kN(x_i|mean_k,variance_k)/ [Σ_l a_lN(x_i|mean_l,variance_l)]

  M(aximization):
  a_k = 1/N Σ_i q_{ik}
  mean_k = Σ_i q_{ik} x_i / Σ_i q_{ik}
  variance_k = Σ_i q_{ik} (x_i-mean_k)^2 / Σ_i q_{ik}

- Always leads to at least a local maximum, convergence very fast in general

Gaussian mixture model

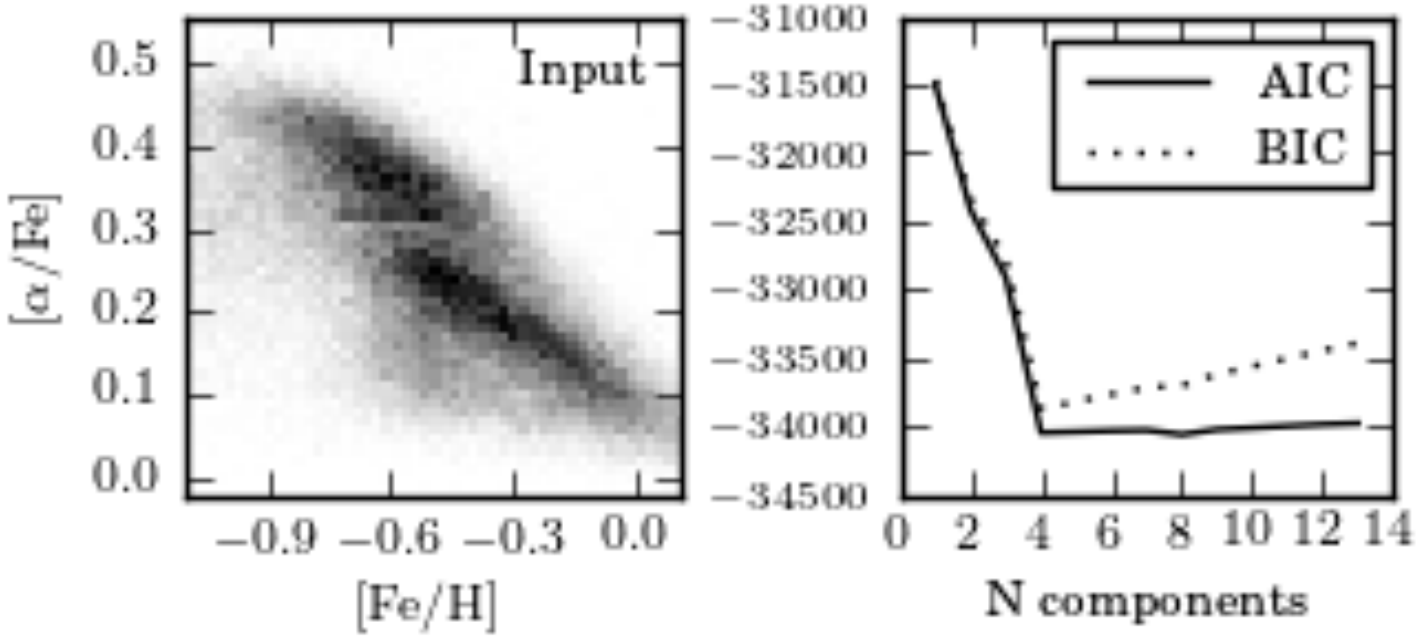
- Parametric, but when *K* is large almost as flexible as a non-parametric model

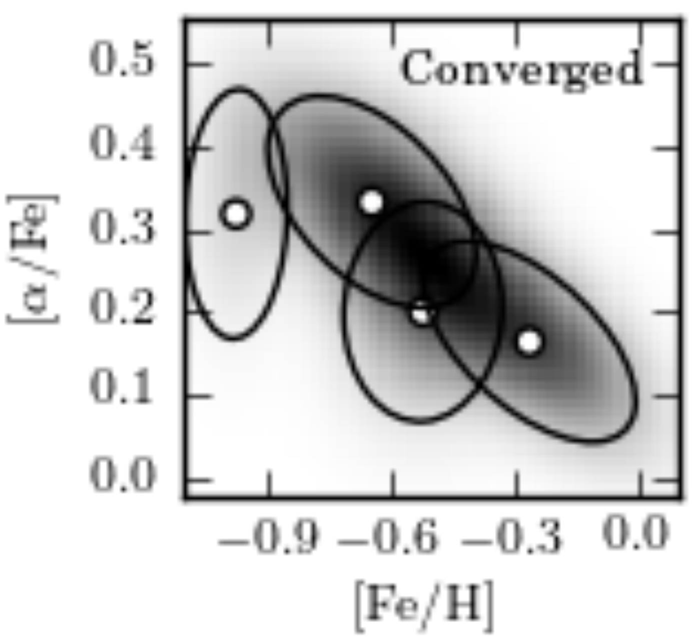
- Need to set *K*, the single hyper-parameter

- Use cross-validation or AIC/BIC

- If you are simply trying to get a good representation of a density, number *K* doesn't matter as long as it's big enough

Example





Ivezic et al. (2014)

- Be careful when interpreting components!!

Gaussian mixtures with errors: extreme deconvolution (XD)

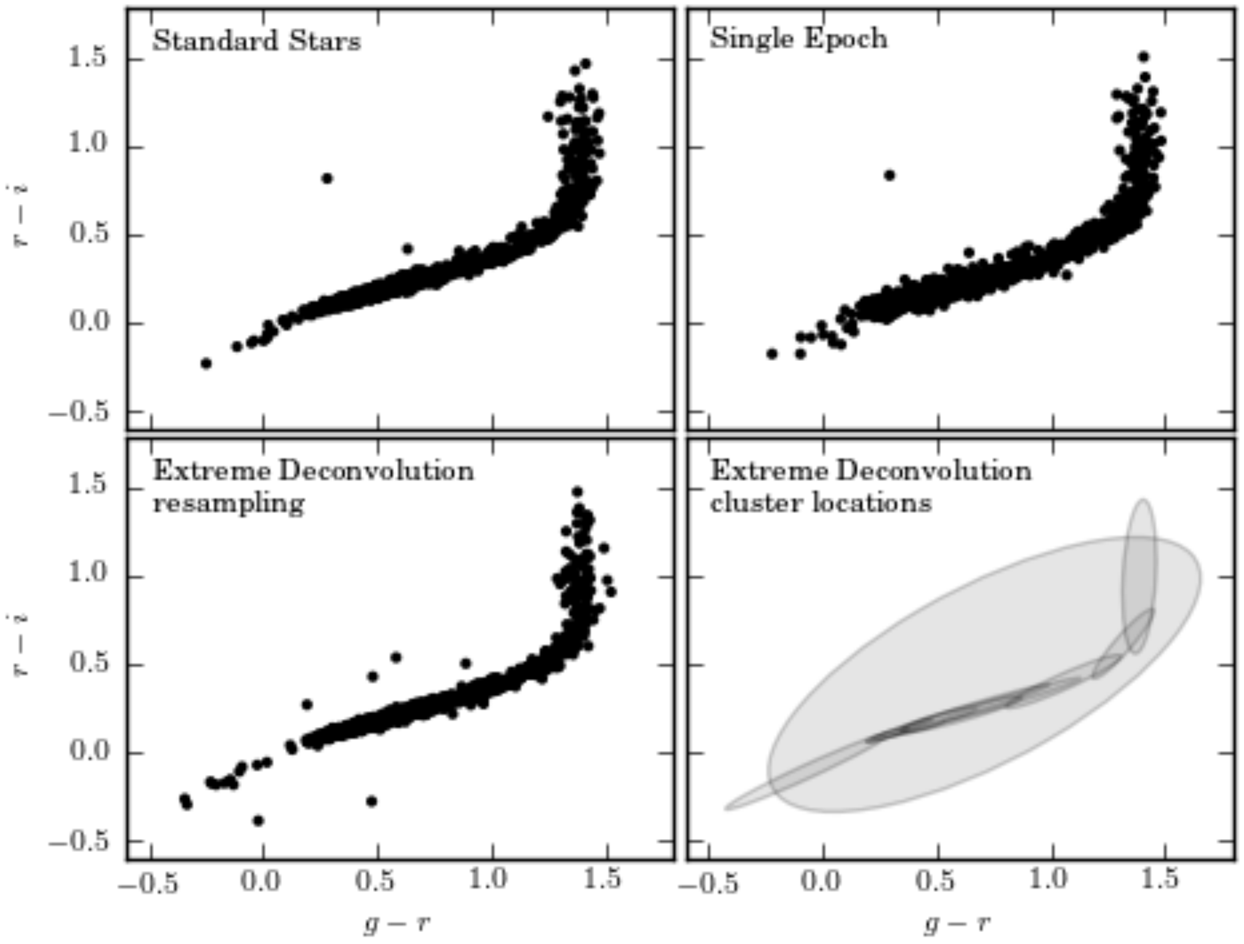
- If data have individual uncertainties (heteroskedastic uncertainties), can still fit a Gaussian mixture model quickly

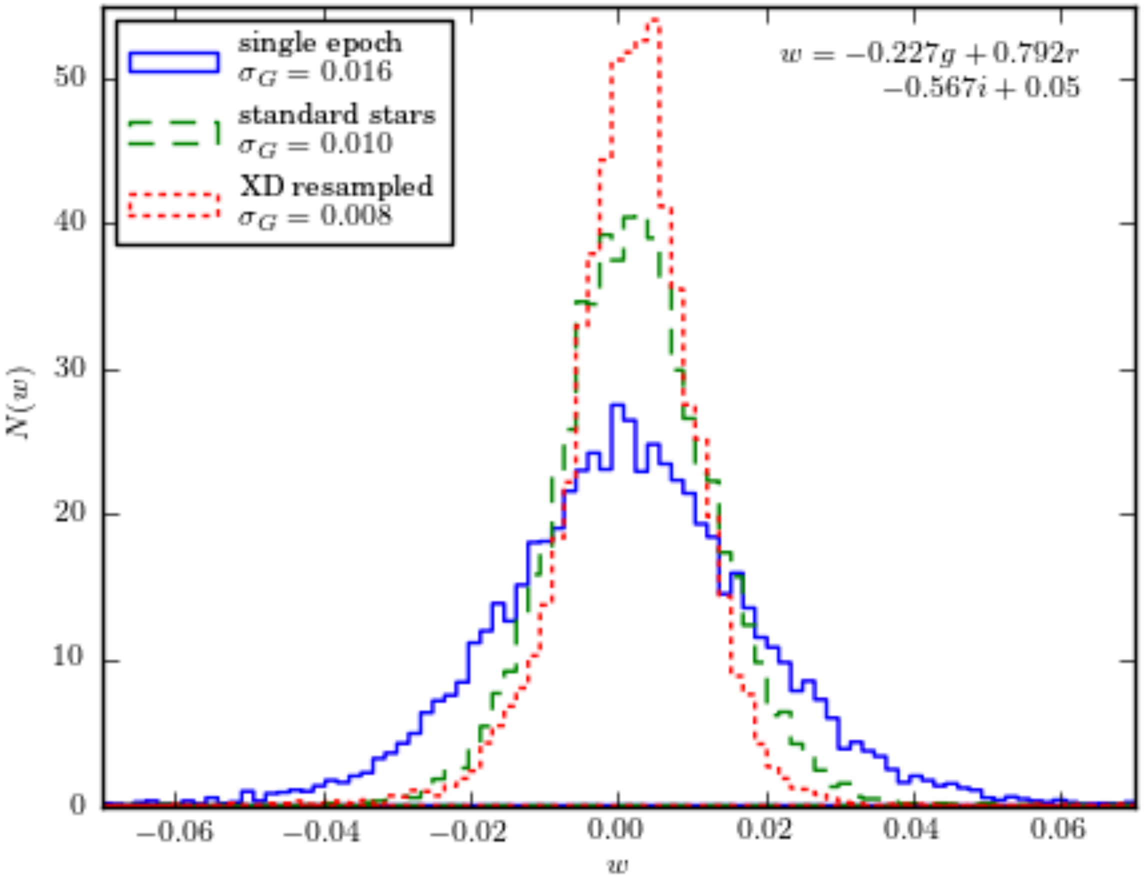
- Trick is to include more *hidden* variables like the q_{ik}: true values x_{ik} if point i was drawn from component k

- Adds a few simple update steps (Bovy et al. 2011)

- Implemented in astroML, fast C version at github/jobovy/extreme-deconvolution

XD example





Ivezic et al. (2014)

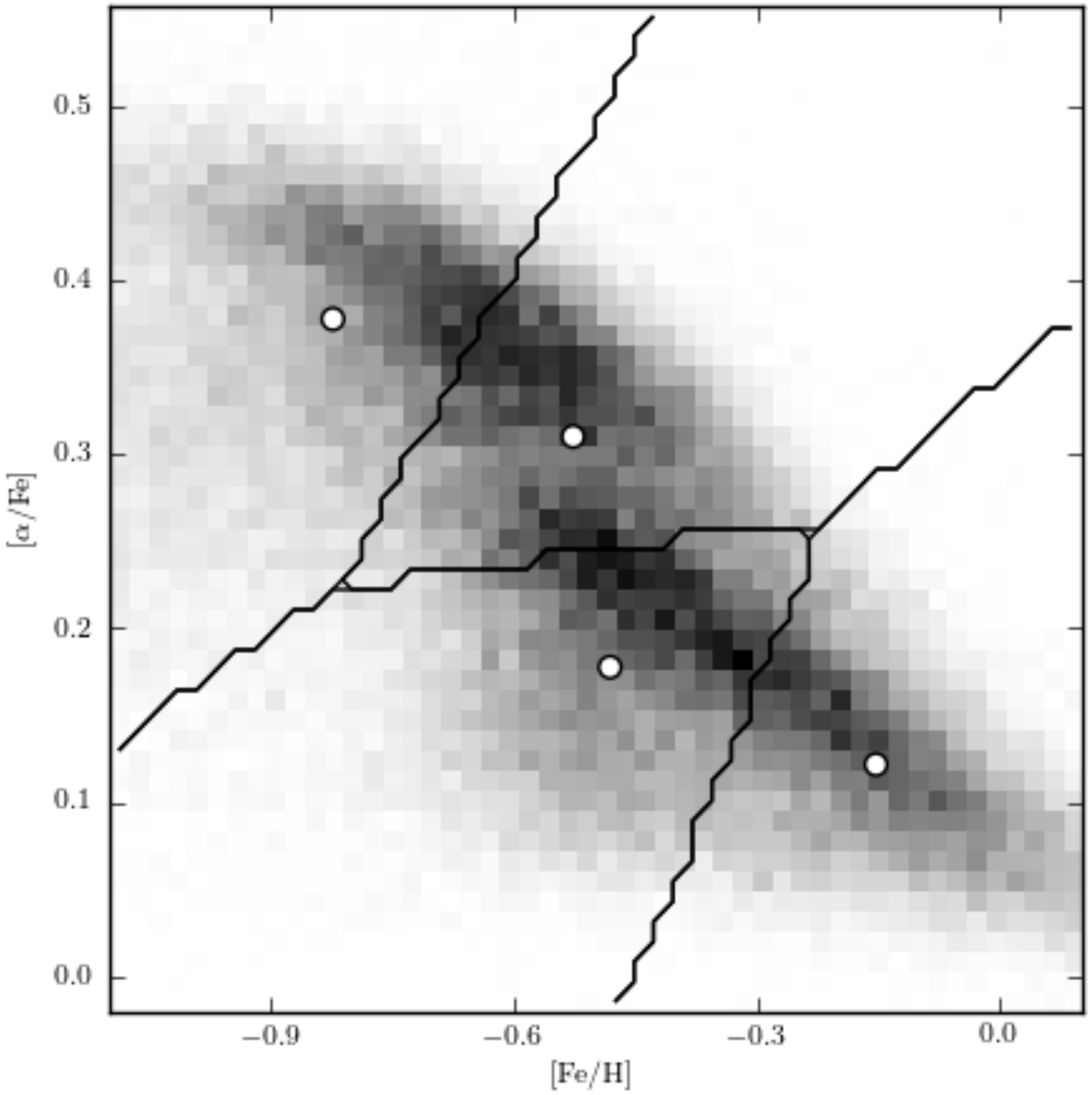
Clustering

- Example of *unsupervised learning*: given set of data x_i, what are the clusters / classes that this data can be divided into?
- Could use a density estimate and find peaks or clearly separated points
- Simplest stand-by algorithm: K-means

K means

- Fix number of clusters *K*
- Optimize $\Sigma_k \Sigma_{i \text{ in } k} |x_i\text{-}m_k|^2$
- Like Gaussian mixture model, but with hard assignments
- Optimization algorithm:
  1. Start with set of $\{m_k\}$
  2. Assign each x_i to its nearest m_k
  3. Compute new m_k as the mean of all of the x_i assigned to cluster k
  4. Go back to 2.
- Could also use medians: K medians

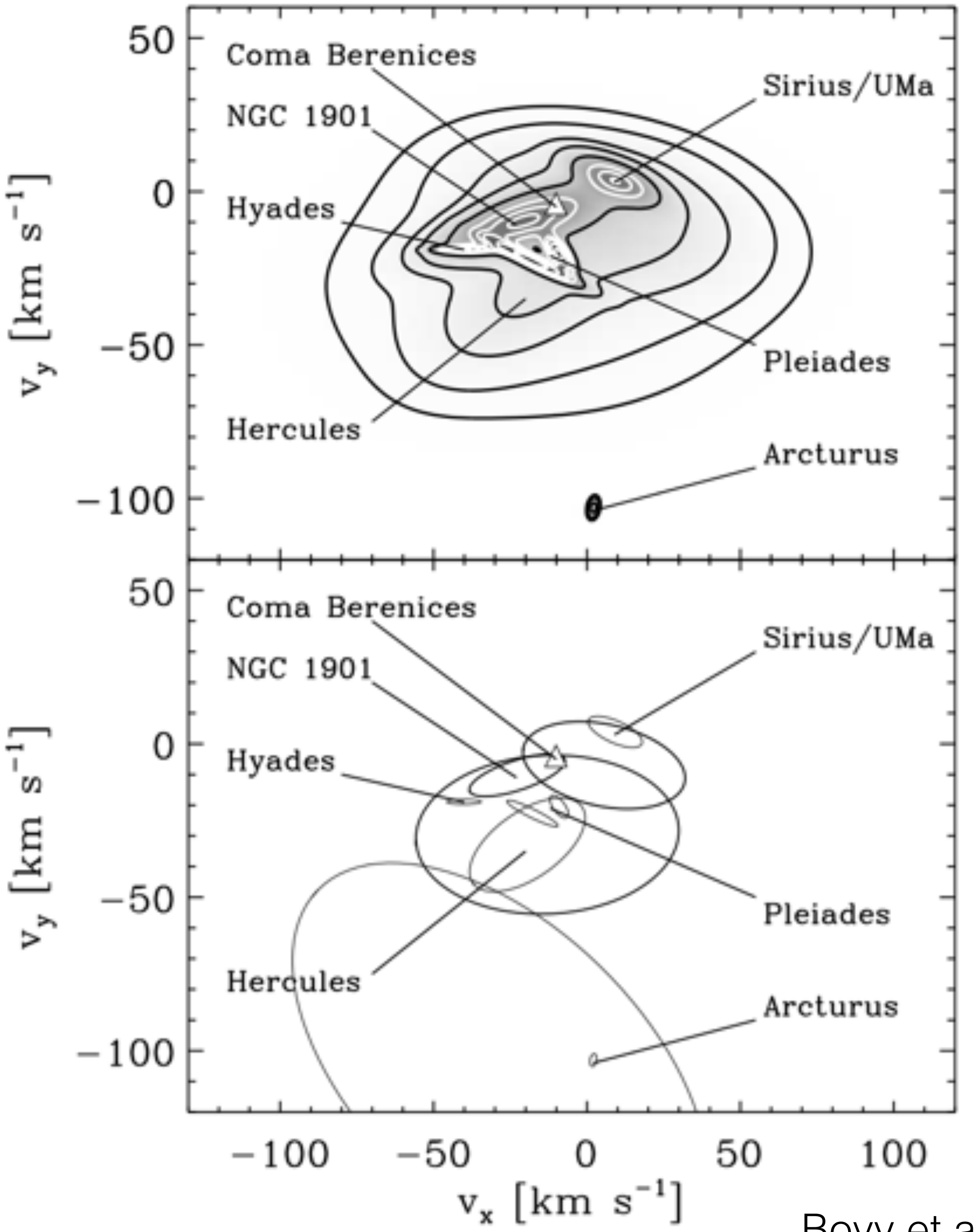
K means example



Ivezic et al. (2014)

Clustering with Gaussian mixtures

- Can work much better because background can be fit out



Bovy et al. (2009)

Procedural clustering

- Gaussian mixture and K-means have the advantage that they optimize an objective function (the likelihood), so the outcome should not depend on how you found the optimal solution

- Procedural clustering defines clusters in a procedural way

- Hierarchical clustering:
  1. Start with N clusters, N=#data
  2. Join two clusters to form N-1 clusters
  3. Repeat

- Join based on: minimum distance between clusters (minimum spanning tree) —> extended clusters, maximum distance between clusters —> compact clusters, friends-of-friends is further example

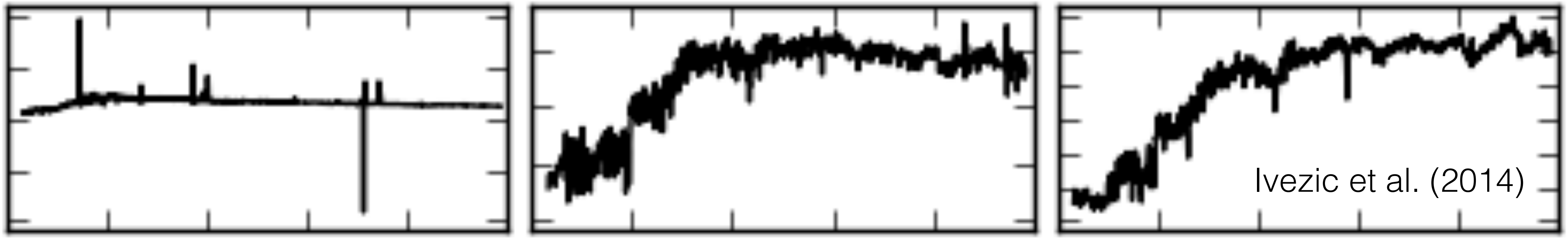
Dimensionality reduction: PCA and ICA

Dimensionality reduction

- Astronomical observations are by their nature high-dimensional

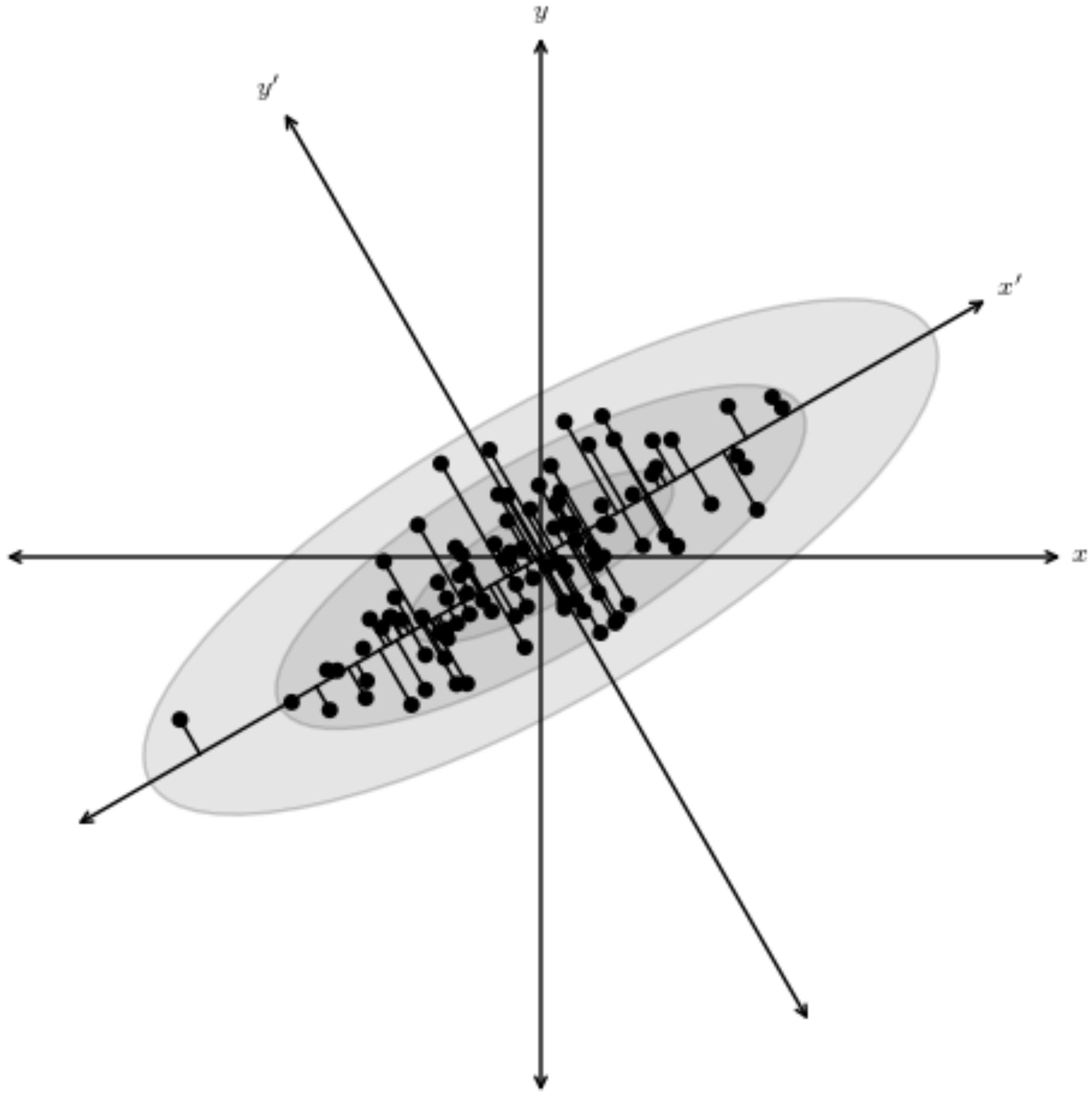
- Need to focus on most important dimensions in the data

- Those dimensions are not necessarily aligned with observed axes, e.g., pixels in a spectrum



Ivezic et al. (2014)

Principal Component Analysis (PCA)



Ivezic et al. (2014)

Principal Component Analysis (PCA)

- Data in D-dimensional space
- Find direction with highest variance
- Rotate such that that direction is x_1
- In the remaining (D-1)-dimensional space do the same: find direction with highest variance, rotate that to x_2
- and so on

PCA using eigenvectors

- Can determine PCA components using eigendecomposition of the data's variance tensor $C_X = X^T X/[N-1]$

- First component r_1 should minimize $r_1^T C_X r_1$ and $|r_1| = 1$: introduce Lagrange multiplier λ_1

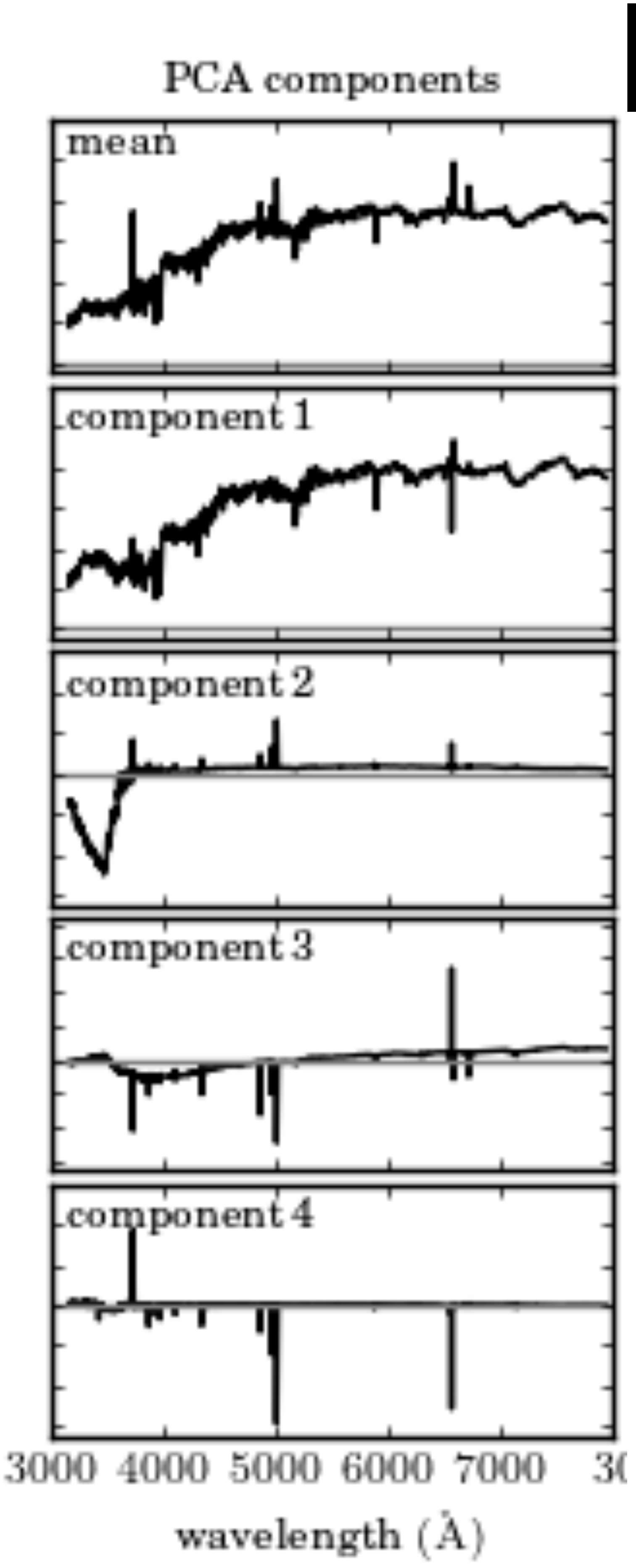
  Minimize $r_1^T C_X r_1 - \lambda_1(r_1^T r_1 - 1)$

  $C_X r_1 - \lambda_1 r_1 = 0$ —> r_1 is an eigenvector of C_X w/ eigenvalue λ_1, must be largest eigenvalue

- Thus, can compute eigendecomposition of C_X, order eigenvectors by their eigenvalues

- In practice, better done with singular-value decomposition

PCA example: galaxy spectra in SDSS



Ivezic et al. (2014)

PCA in practice

- Because you are rotating, technically only applies when all dimensions have the same units

- If you want to apply PCA to dimensions with different units, need to divide out the units: subtract the mean and divide by typical value or ‘whiten’ by subtracting the mean and dividing by the data’s standard deviation

- If data have errors, need to account for this; if they are different for different dimensions and/or data points, need to solve for PCA components iteratively

Dimensionality reduction with PCA

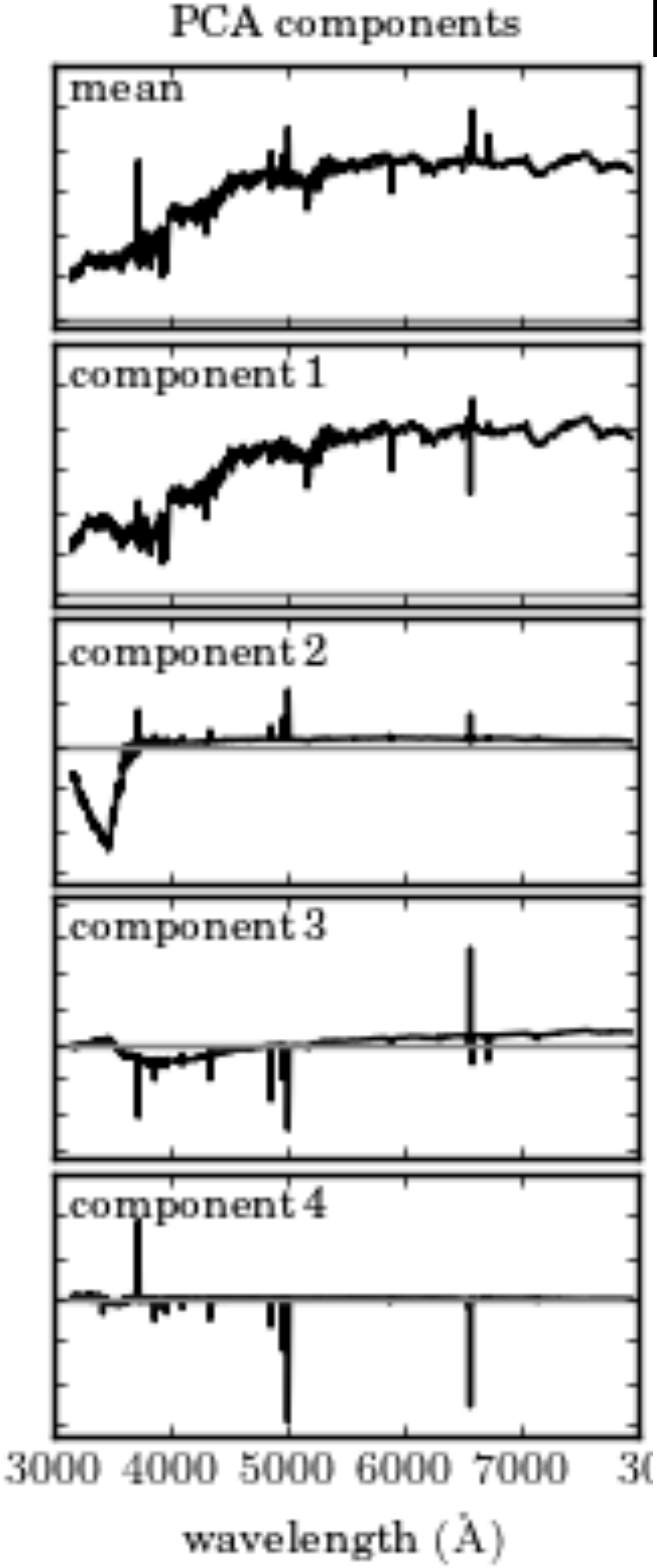
- PCA decomposition tells you which directions explain most of the variation in the data

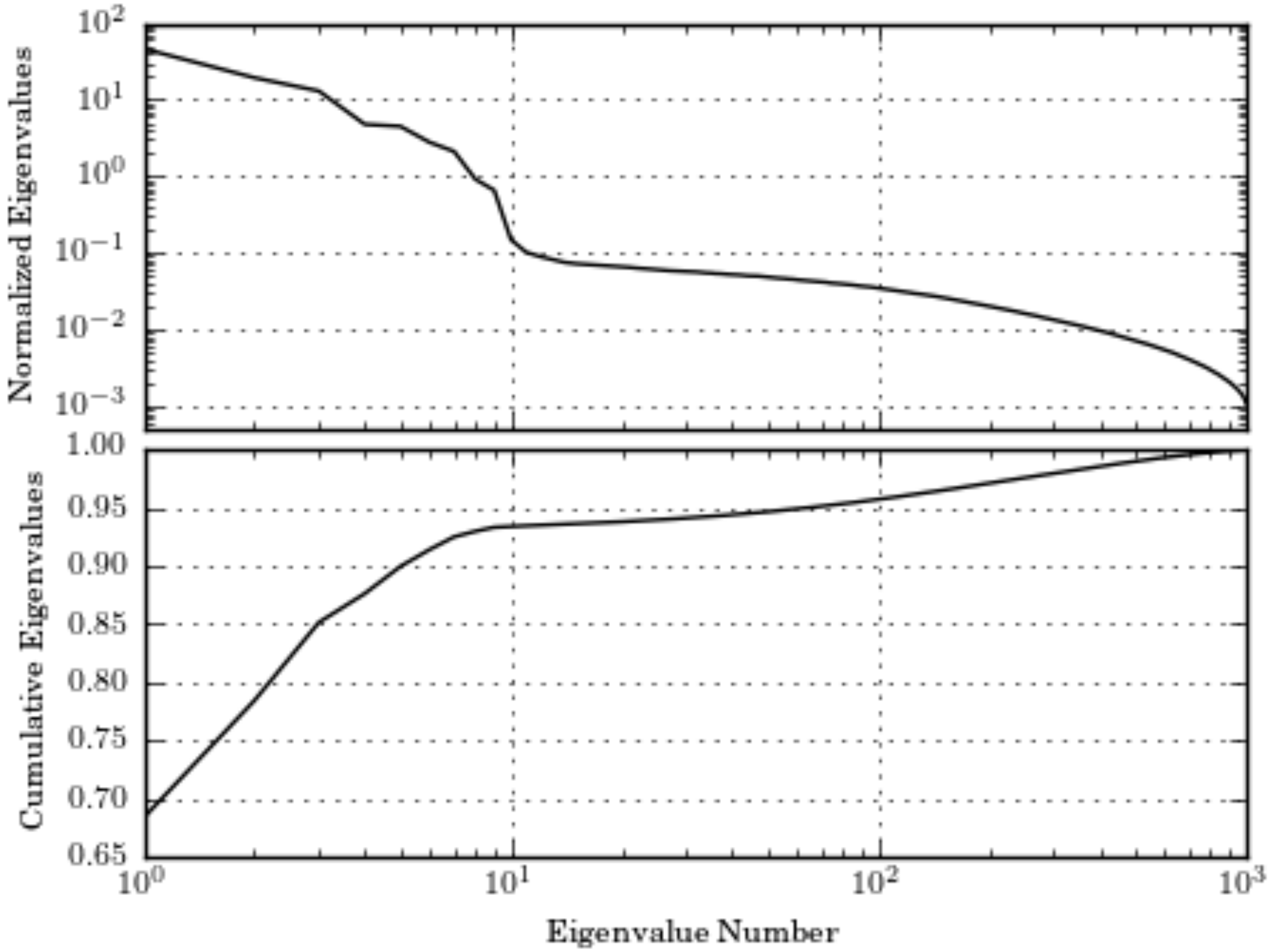
- Can cut at a certain number K <= D of PCA components that explain X% of the variance (K=D explains 100%)

- If K << D, can significantly reduce the dimensionality of the data

- Where to cut? Compare to expected noise level, or decide how much variance you want to explain, search for features in the (explained-variance) vs. K plot

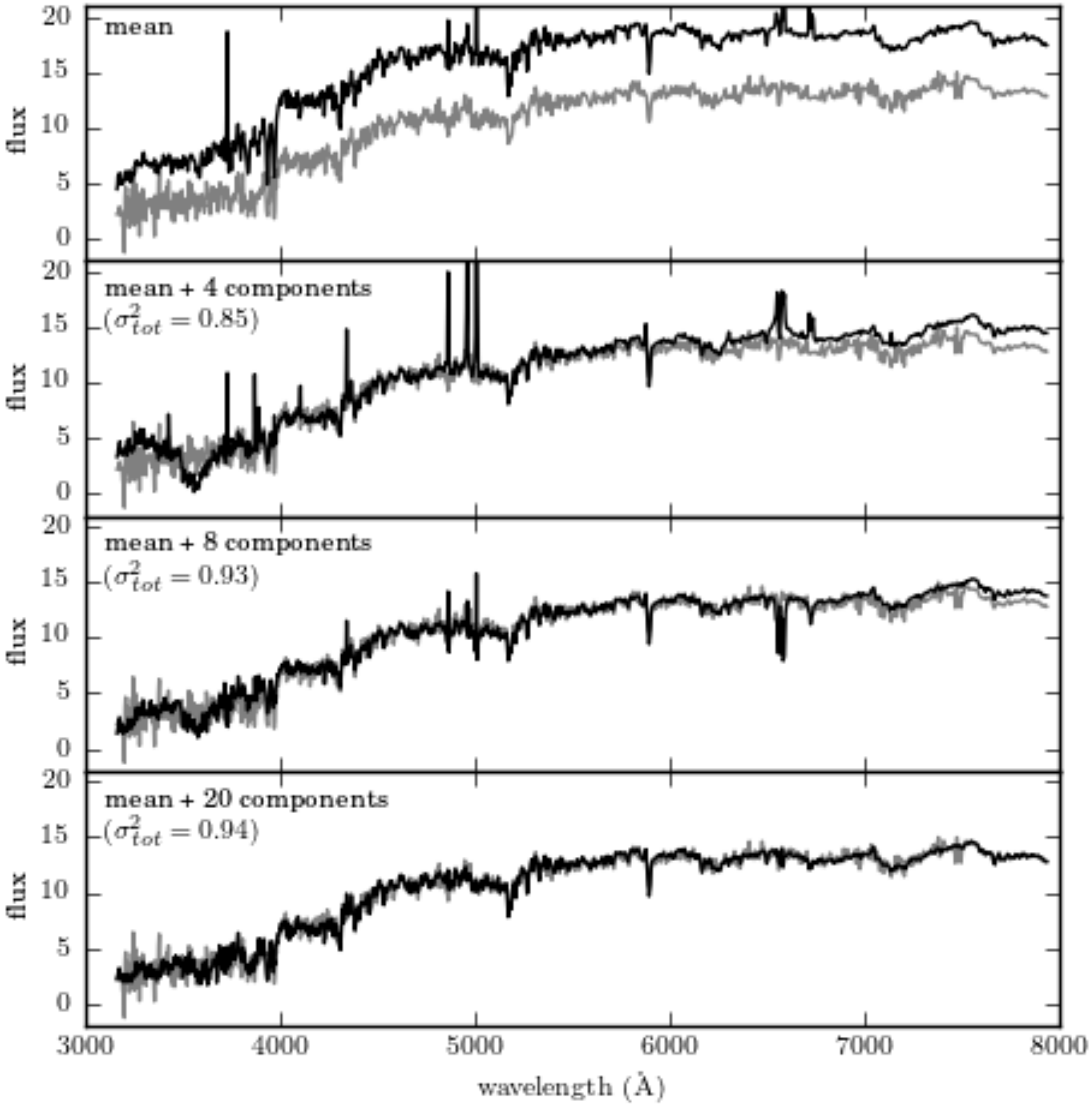
PCA example: galaxy spectra in SDSS





Ivezic et al. (2014)

PCA example: galaxy spectra in SDSS

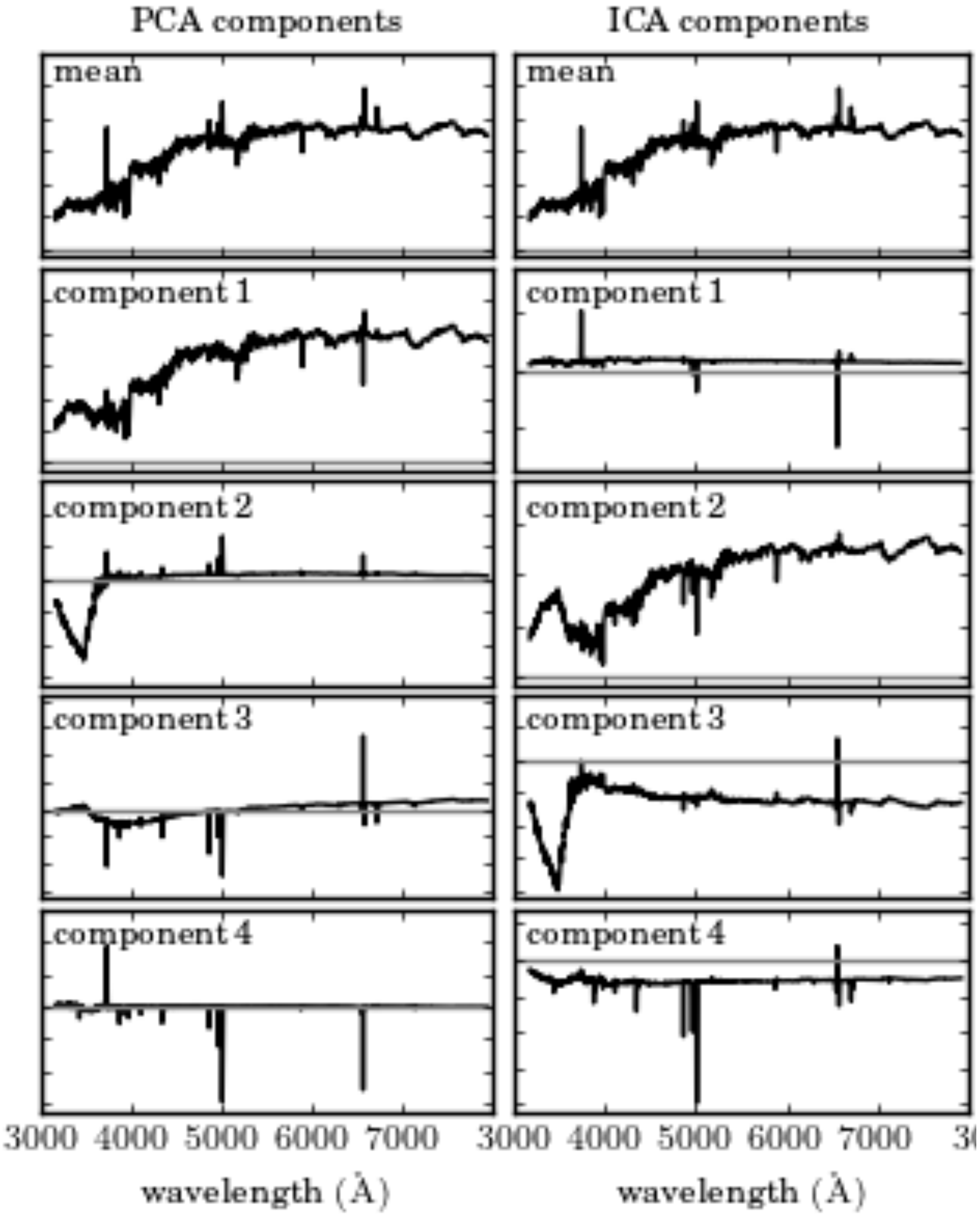


Independent Component Analysis (ICA)

- Generalization of PCA
- Find directions in high-dimensional space, such that each direction's data distribution is statistically independent:

  $f(x^p, y^q) = f(x^p)\ f(y^q)$ for some p,q
- p=q=1: PCA (requires uncorrelated data)
- In general: maximize non-Gaussianity of individual distributions f(x): kurtosis, negative entropy

ICA example: galaxy spectra in SDSS

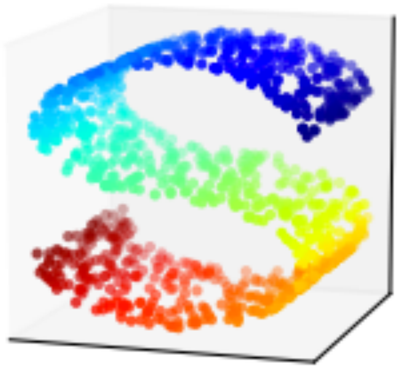


Ivezic et al. (2014)

Other dimensionality reduction techniques

- Non-negative matrix factorization: similar to PCA/ICA, but components are always positive

- Manifold learning, e.g., locally-linear embedding: can deal with complex lower-dimensional objects in higher-dimensional space



- t-SNE: t-distributed stochastic neighbor embedding: models high-dimensional space as 2D in such a way that points close in high-D are close in 2D and points far are far in both

Regression

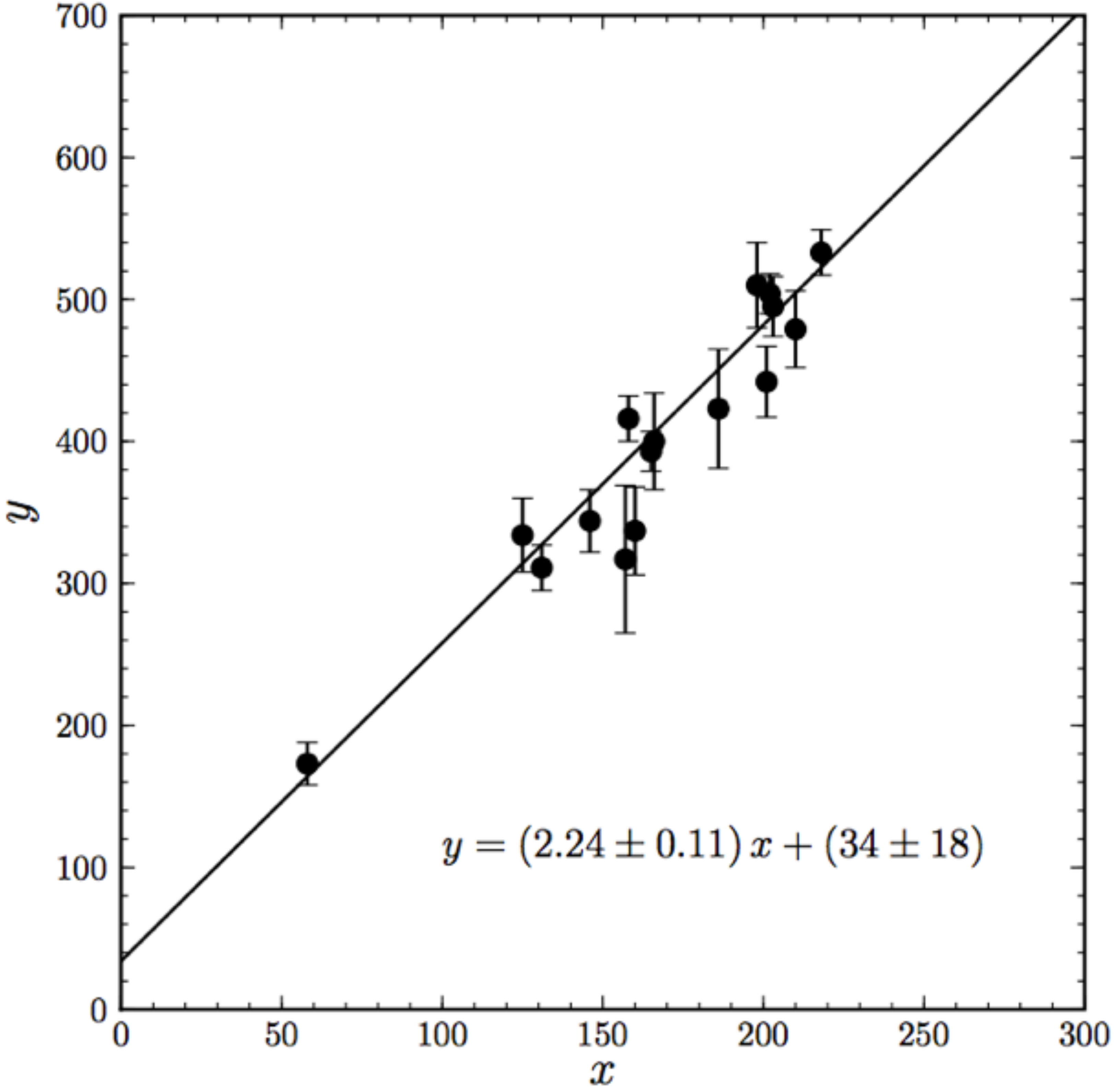
Regression problems

- Have data set (x,y) —> y(x)?
- Issues:
  - y has errors with known Gaussian distribution, can be different
  - y has errors with known non-Gaussian distribution
  - y has unknown errors
  - x and y have known Gaussian errors
  - x and y have unknown errors
- Model complexity:
  - Linear —> relatively easy
  - Non-linear —> hard!

Regression: straight line

- Model is $y = mx + b$
- Maximizing likelihood equivalent to solving:

  $Y = A X$, with $Y^T = [y_0, y_1, \ldots, y_{N-1}]$, $X^T = [m, b]$,
  $A = [[x_0, 1], [x_1, 1], \ldots, [x_{N-1}, 1]]$
- No errors: $X = [A^T A]^{-1} A^T Y$
- With errors: $C = [[\sigma^2{}_0, 0, 0, \ldots, 0], [0, \sigma^2{}_1, 0, \ldots, 0], \ldots, [0, \ldots, 0, \sigma^2{}_{N-1}]]$:
  $X = [A^T C^{-1} A]^{-1} A^T C^{-1} Y$
- Prediction for x_{new}: $[x_{new}, 1] \times [[A^T C^{-1} A]^{-1} A^T C^{-1} Y]$


700
600
500
400
300
200
100
0
y
0
50
100
150
200
250
300
x
$y = (2.24 \pm 0.11)\,x + (34 \pm 18)$

Regression: basis function fitting

- Higher-order polynomials: $y = c\,x^2 + m\,x + b$
- Proceed the same way, only thing different is *design matrix* A
- $Y = A\,X$, with $Y^T = [y_0, y_1, \ldots, y_{N-1}]$, $X^T = [c, m, b]$,
  $A = [[x^2_0, x_0, 1], [x^2_1, x_1, 1], \ldots, [x^2_{N-1}, x_{N-1}, 1]]$
- No errors: $X = [A^T A]^{-1} A^T Y$
- With errors: $C = [[\sigma^2_0, 0, 0, \ldots, 0], [0, \sigma^2_1, 0, \ldots, 0], \ldots, [0, \ldots, 0, \sigma^2_{N-1}]]$:
  $X = [A^T C^{-1} A]^{-1} A^T C^{-1} Y$

Regression: basis functions

- Can use many more basis functions and approach non-parametric regression

- E.g., Gaussian, piecewise-polynomial

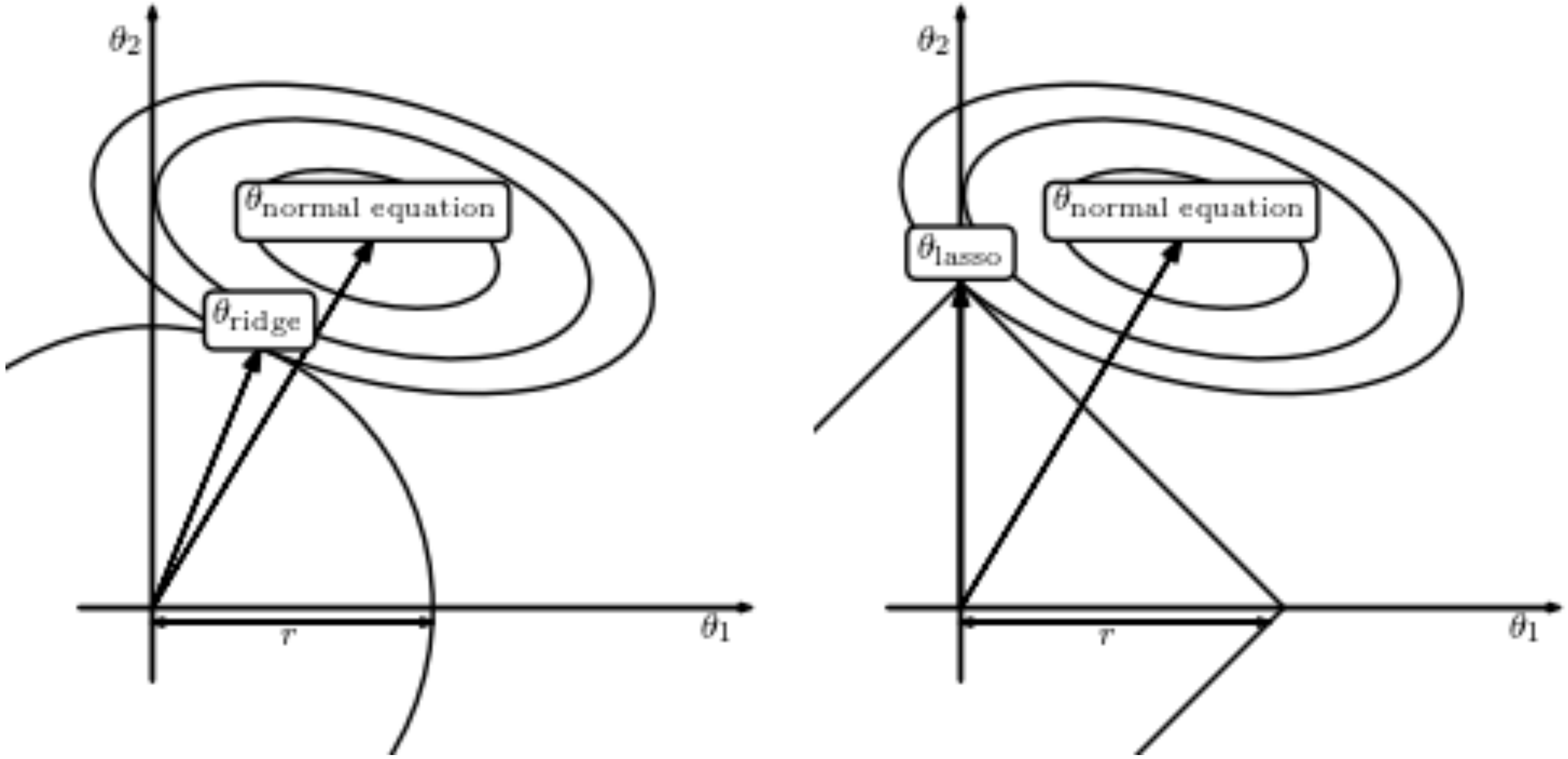
- # of parameters grows —> need to penalize complexity

- Maximize: log L + regularization term

- regularization term:

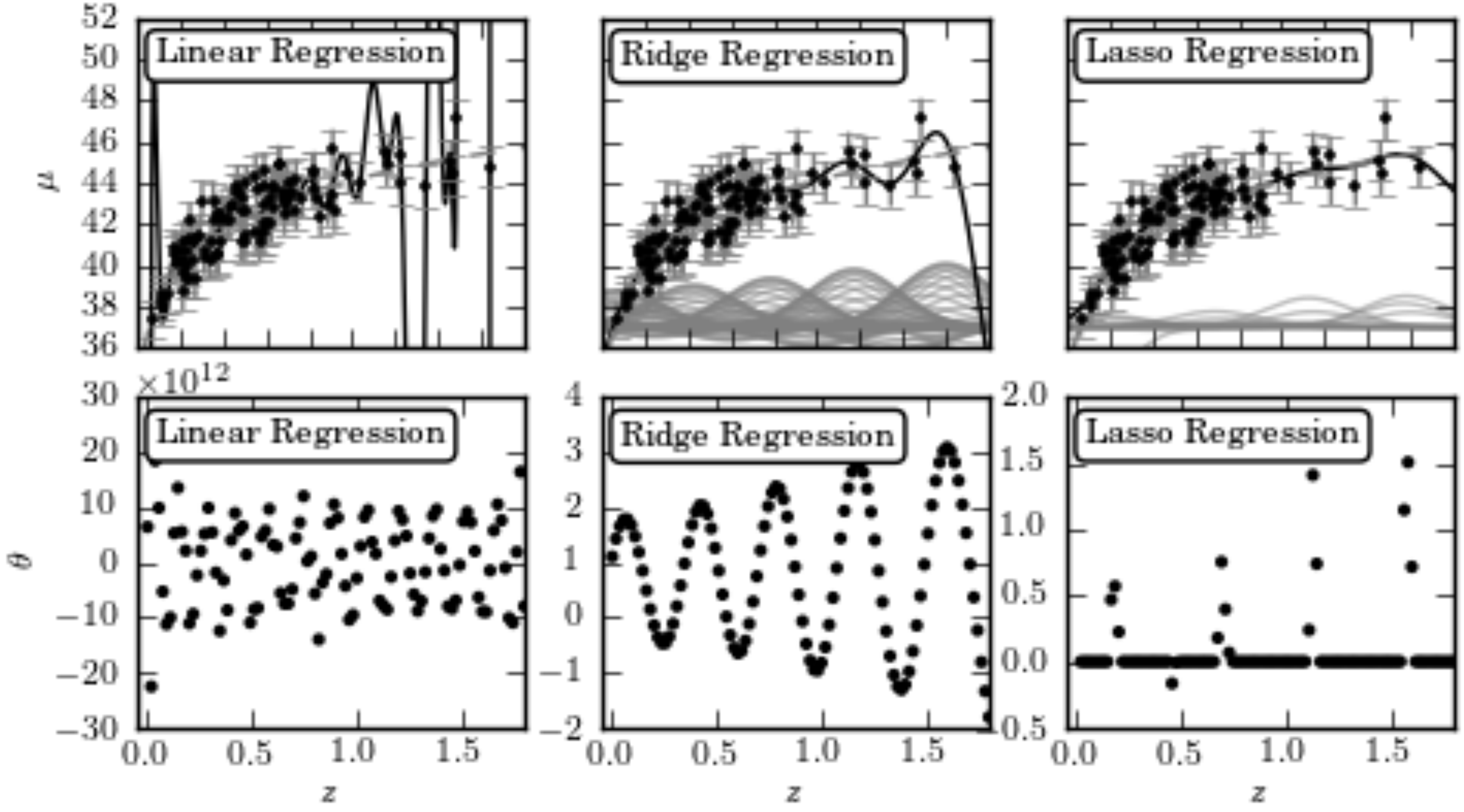
  $\lambda \int dx|y''(x)|^2$ —> spline
  $\lambda\ |X^TX|$ —> ridge regression
  $\lambda\ |X|$ —> LASSO regression (prefers $X = 0$)

- Need to set λ —> cross-validation etc.


θ_2
$\theta_{\mathrm{normal\ equation}}$
θ_{ridge}
r
θ_1
θ_2
θ_{lasso}
$\theta_{\mathrm{normal\ equation}}$
r
θ_1

Ivezic et al. (2014)

Basis function regression example



Ivezic et al. (2014)

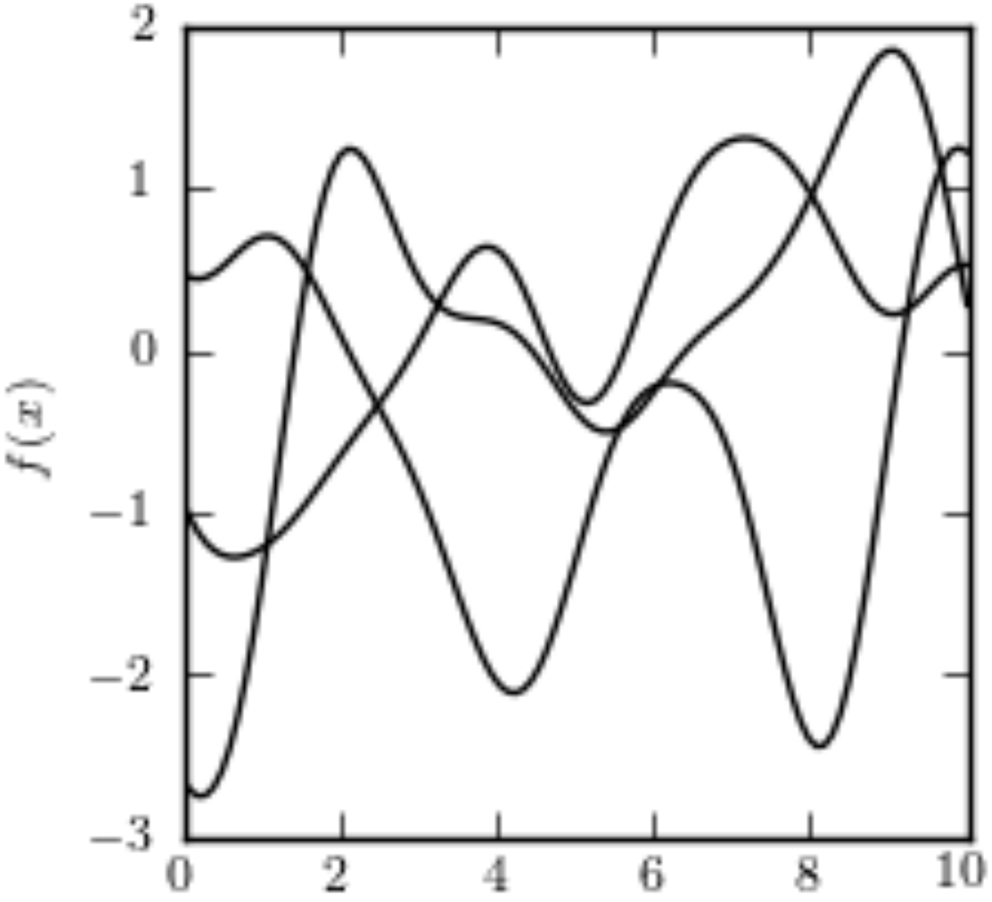
Gaussian Processes (GP)

- Gaussian process is an example of an infinite-dimensional model, sets a prior on functions
- GP: joint distribution of any $[y(x_0),y(x_1),..,y(x_{N-1})]$ is Gaussian
- GP: characterized by mean function m(x) and covariance function $\mathrm{Cov}(x_1,x_2)$ that specify this joint distribution
- Mean and covariance function characterize by hyperparameters
- Magic of Gaussians make everything easy to deal with

Gaussian Processes (GP)

- Need to choose Cov(x_1,x_2), popular choice is σ^2 x exp(-(x_1-x_2)^2/[2h^2]) with parameters σ and h

- Can then draw functions from this Gaussian

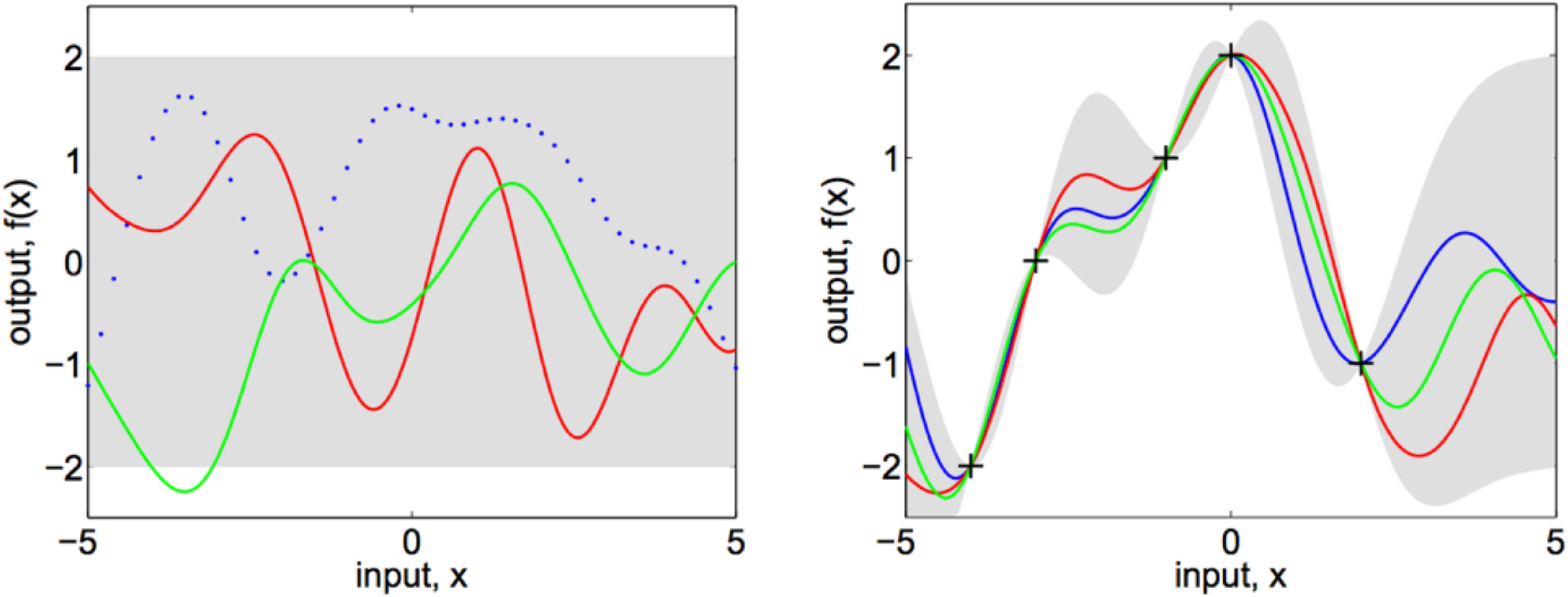


Ivezic et al. (2014)

Gaussian Processes (GP)

- If you have some observed data (x_i,y_i) with error bars, can write down the joint distribution of $[x_{0,new},x_{1,new},\ldots,x_{K-1,new},x_{i0},x_{i1},\ldots,x_{1N-1}]$ and condition on $x_{0,new},x_{1,new},\ldots,x_{K-1,new}$
- This gives the posterior distribution over functions, which is still Gaussian

GP example



(a), prior

(b), posterior

Rasmussen & Williams (2006)

GP math

- Joint distribution

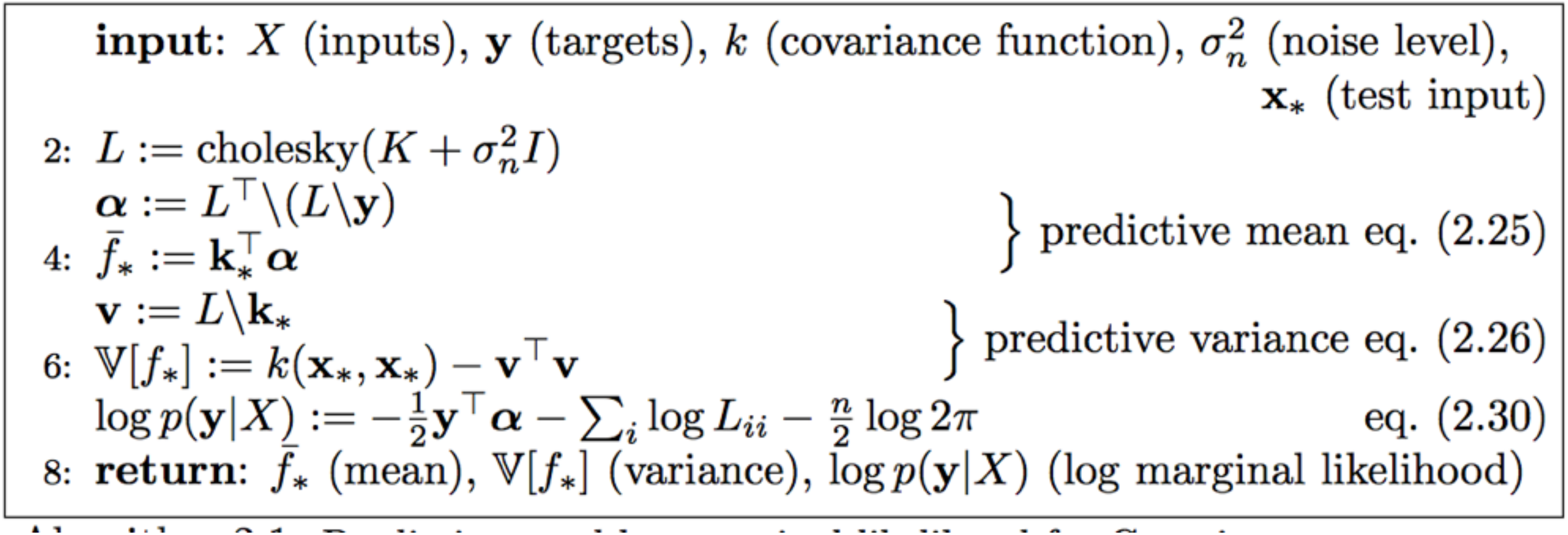
$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X,X)+\sigma_n^2 I & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) \end{bmatrix}\right).$$

- Conditioning on observed points f

$$\begin{aligned} \mathbf{f}_*|X,\mathbf{y},X_* \;&\sim\; \mathcal{N}\big(\bar{\mathbf{f}}_*,\ \mathrm{cov}(\mathbf{f}_*)\big), \text{ where} \\ \bar{\mathbf{f}}_* \;&\triangleq\; \mathbb{E}[\mathbf{f}_*|X,\mathbf{y},X_*] \;=\; K(X_*,X)[K(X,X)+\sigma_n^2 I]^{-1}\mathbf{y}, \\ \mathrm{cov}(\mathbf{f}_*) &= K(X_*,X_*) - K(X_*,X)[K(X,X)+\sigma_n^2 I]^{-1}K(X,X_*). \end{aligned}$$

Rasmussen & Williams (2006)

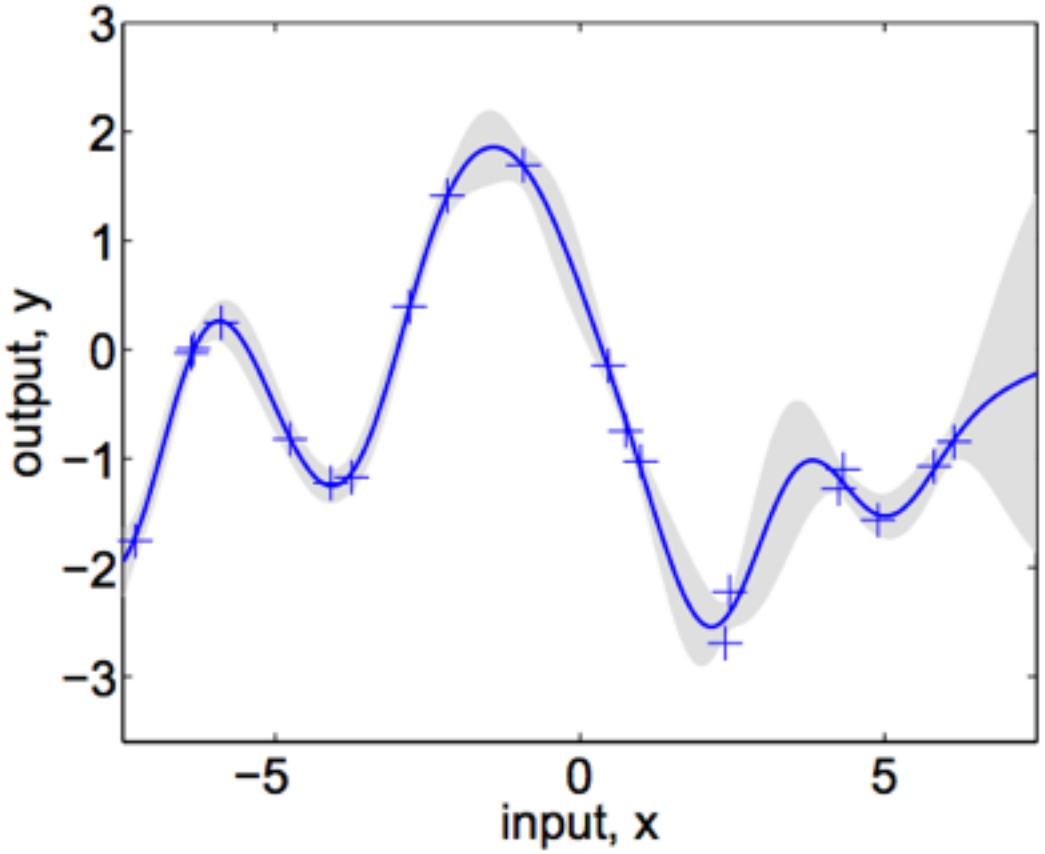
GP algorithm



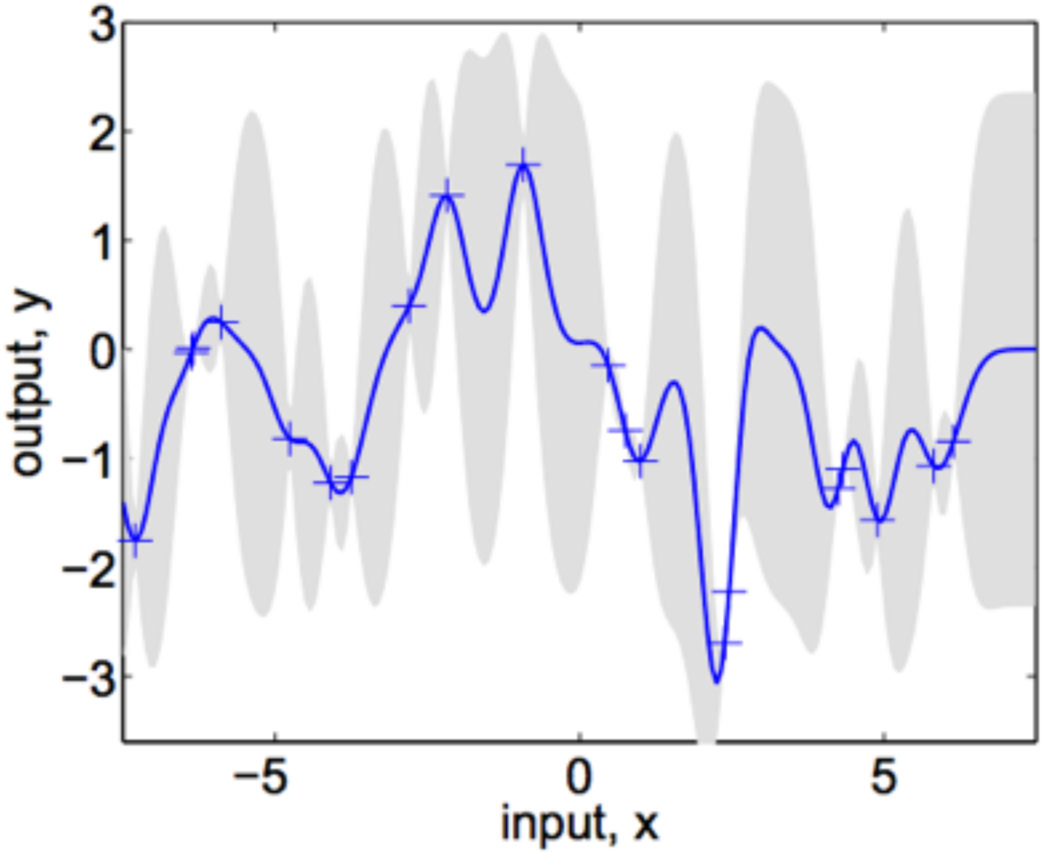
```
input: $X$ (inputs), $\mathbf{y}$ (targets), $k$ (covariance function), $\sigma_n^2$ (noise level), $\mathbf{x}_*$ (test input)
$L := \mathrm{cholesky}(K + \sigma_n^2 I)$
$\boldsymbol{\alpha} := L^\top \backslash (L \backslash \mathbf{y})$
                                                  } predictive mean eq. (2.25)
$\bar{f}_* := \mathbf{k}_*^\top \boldsymbol{\alpha}$
$\mathbf{v} := L \backslash \mathbf{k}_*$
                                                  } predictive variance eq. (2.26)
$\mathbb{V}[f_*] := k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{v}^\top \mathbf{v}$
$\log p(\mathbf{y}|X) := -\frac{1}{2}\mathbf{y}^\top \boldsymbol{\alpha} - \sum_i \log L_{ii} - \frac{n}{2} \log 2\pi$      eq. (2.30)
return: $\bar{f}_*$ (mean), $\mathbb{V}[f_*]$ (variance), $\log p(\mathbf{y}|X)$ (log marginal likelihood)
```

Rasmussen & Williams (2006)

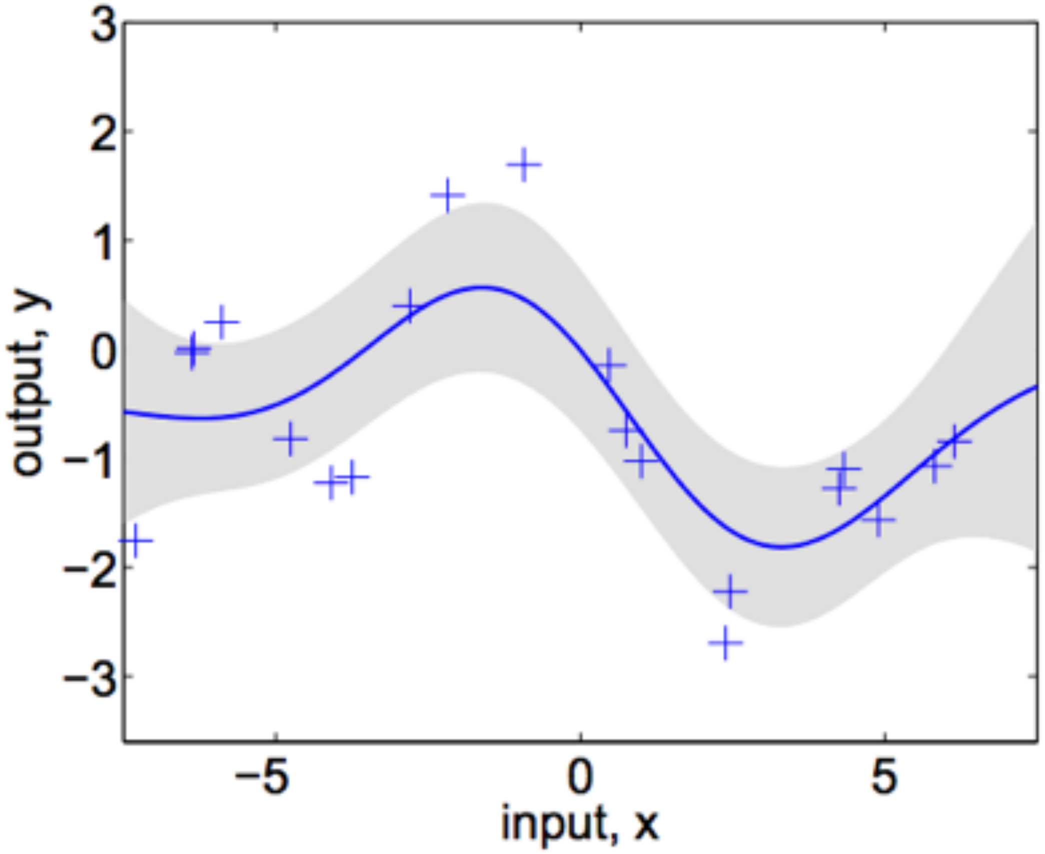
Hyper-parameters



(a), $\ell = 1$

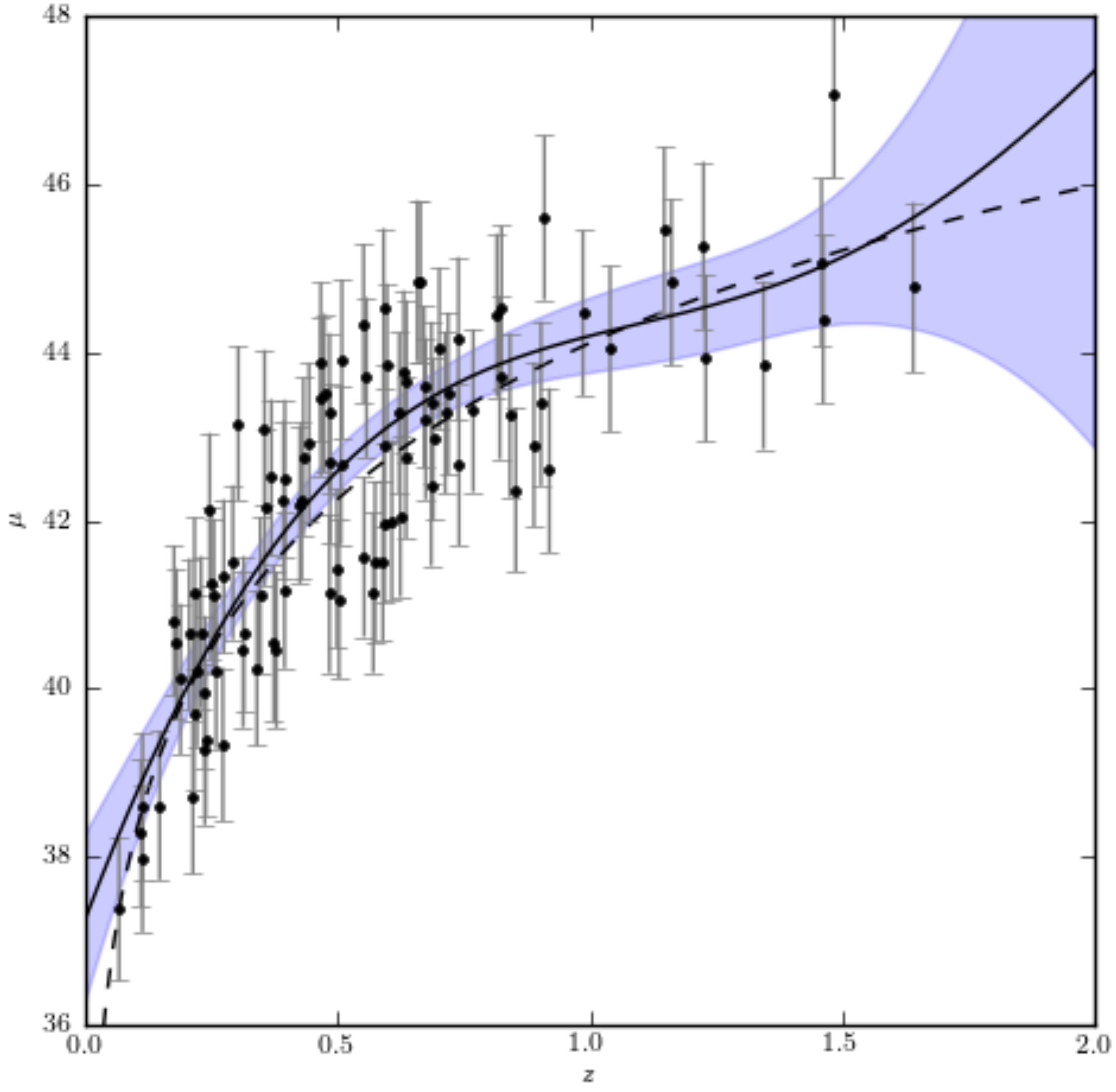


(b), $\ell = 0.3$



(c), $\ell = 3$

Another GP example



Ivezic et al. (2014)

Classification

Classification

- Example of *supervised learning*

- Have training data set of attributes x_i with labels for K classes

- Learn how to assign labels based on attributes to classify unknown sources

- Example: (u,g,r,i,z) —> (quasar,star,galaxy)

Classification metrics

- Purity: fraction of objects assigned to class k that truly are part of class k
- Completeness: fraction of true class-k objects that is assigned to class k
- Difficult to maximize both!

Classification using density estimation

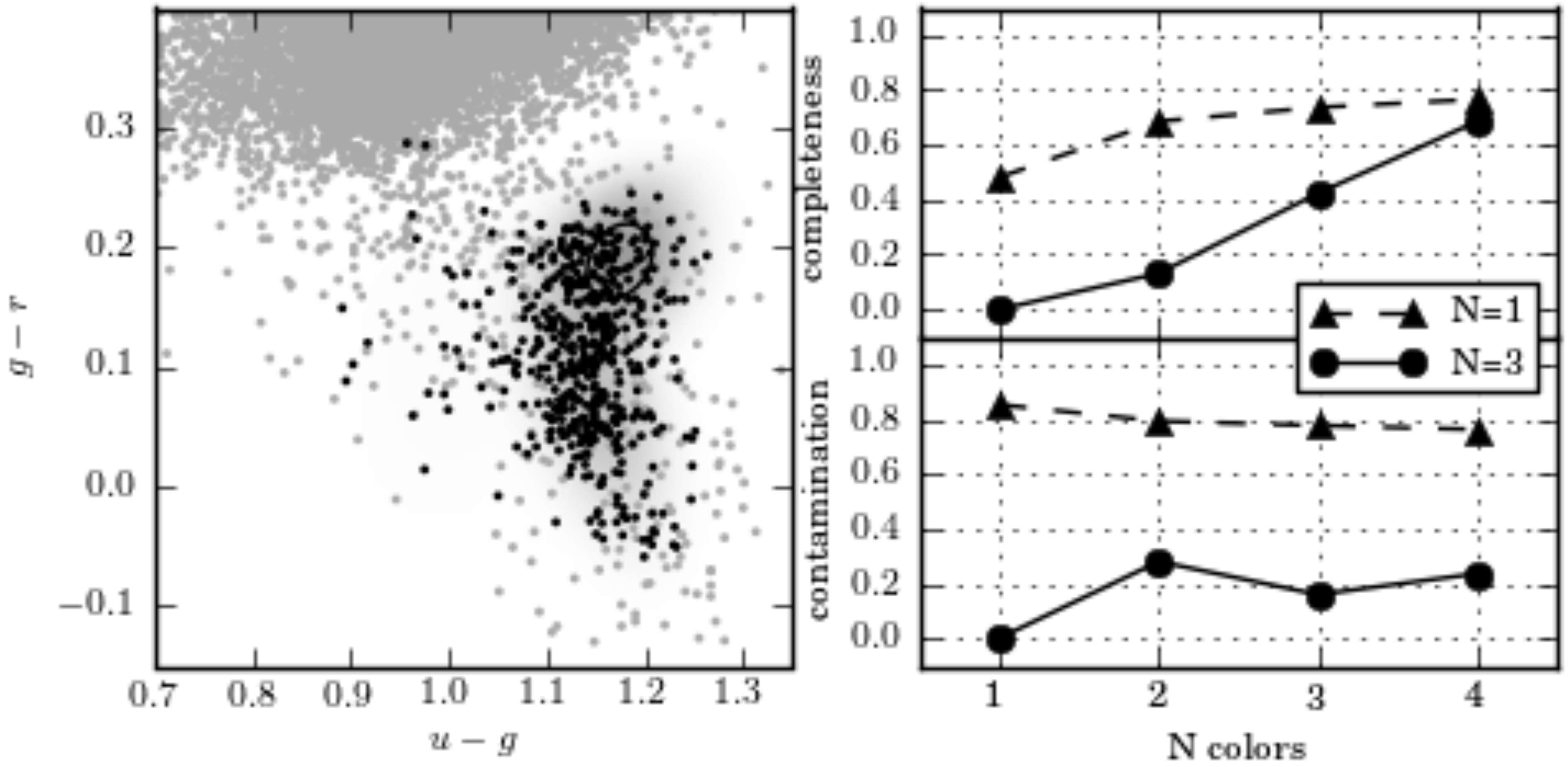
- Can estimate densities for each class $\rho_k(x) = p(x_{new}|\text{class } k)$ using density-estimation techniques discussed earlier

- Assign new classes using Bayes theorem:

$$p(\text{class } m|x_{new}) = \frac{p(x_{new}|\text{class } m)\ p(\text{class } m)}{\Sigma_k\ p(x_{new}|\text{class } k)\ p(\text{class } k)}$$

- Allows for full power of density estimation

Example with Gaussian mixtures



Ivezic et al. (2014)

Non-parametric classification: k-nearest neighbor

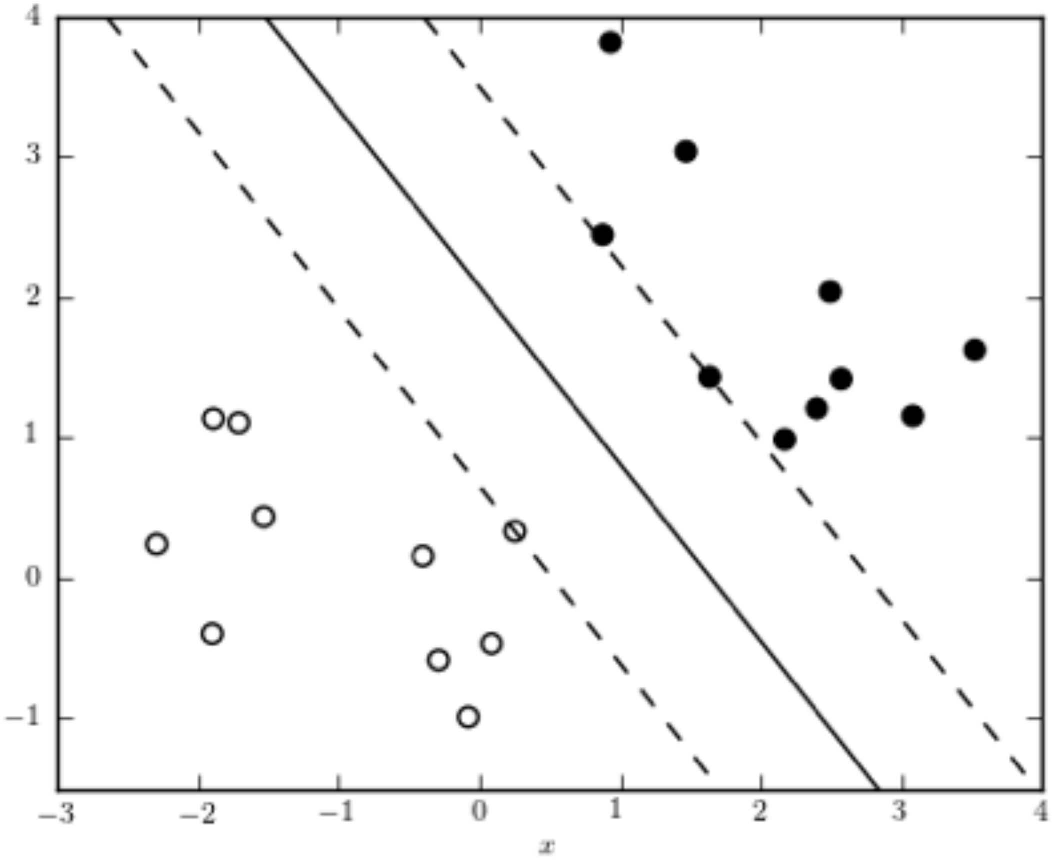
- Simple: Look at the k nearest neighbors in the training set —> assign class based on consensus

- Requires:
  - Distance function
  - Consensus building: can assign weights to neighbors based on, e.g., the distance

- Expensive for large training sets (always need to consider all data)

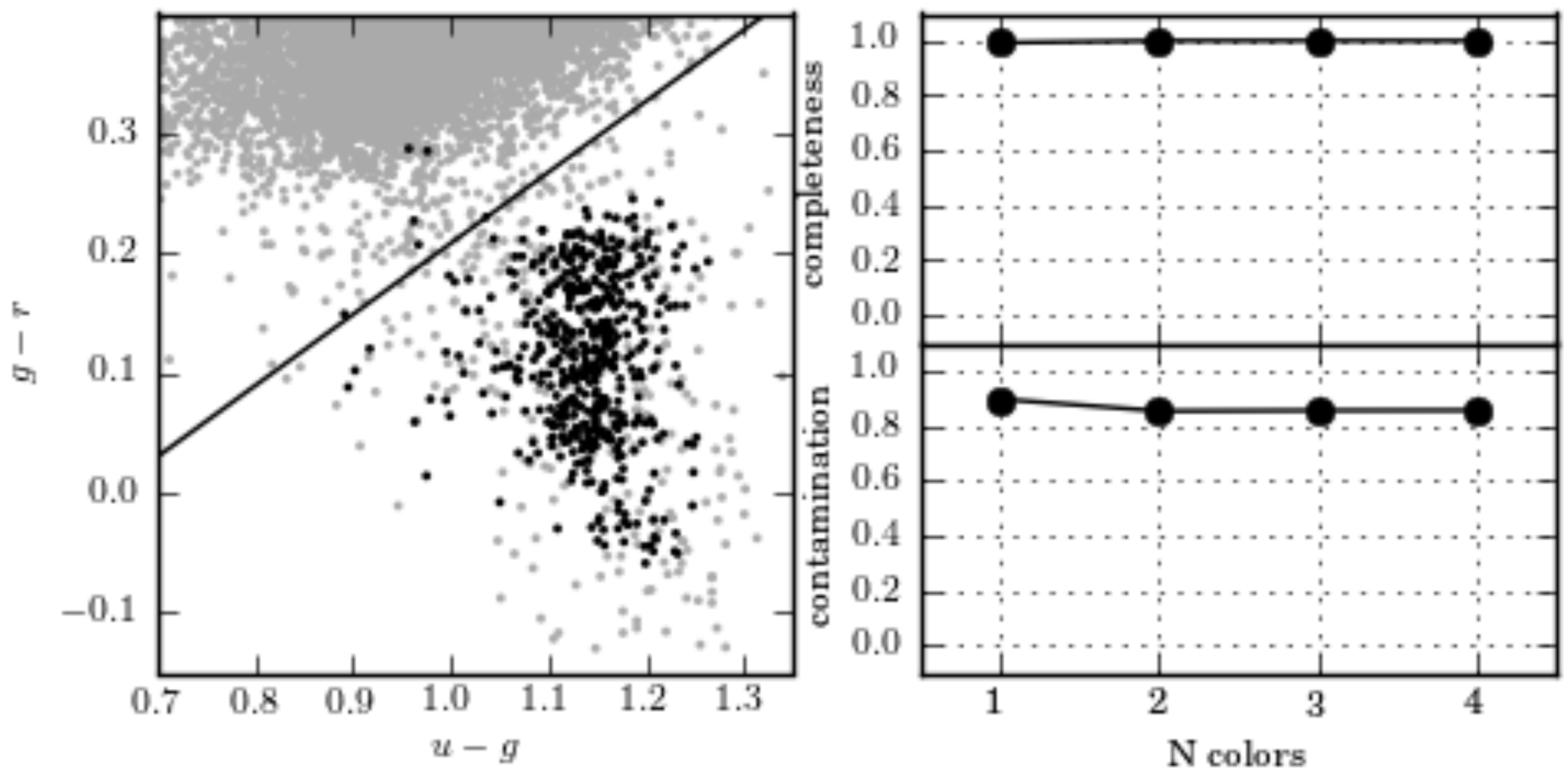
Support Vector Machines

- Find hyperplane in **x** that maximizes the distance between two classes
- That hyperplane is entirely described by the points that lie on it —> support vectors
- Labels y={-1,1}, hyperplane: minimize |m| subject to $y_i(b+mx_i)$ >= 1 for all i
- Can add loss function proportional to distance if data cannot be separated —> hyperparameter



Ivezic et al. (2014)

SVM example

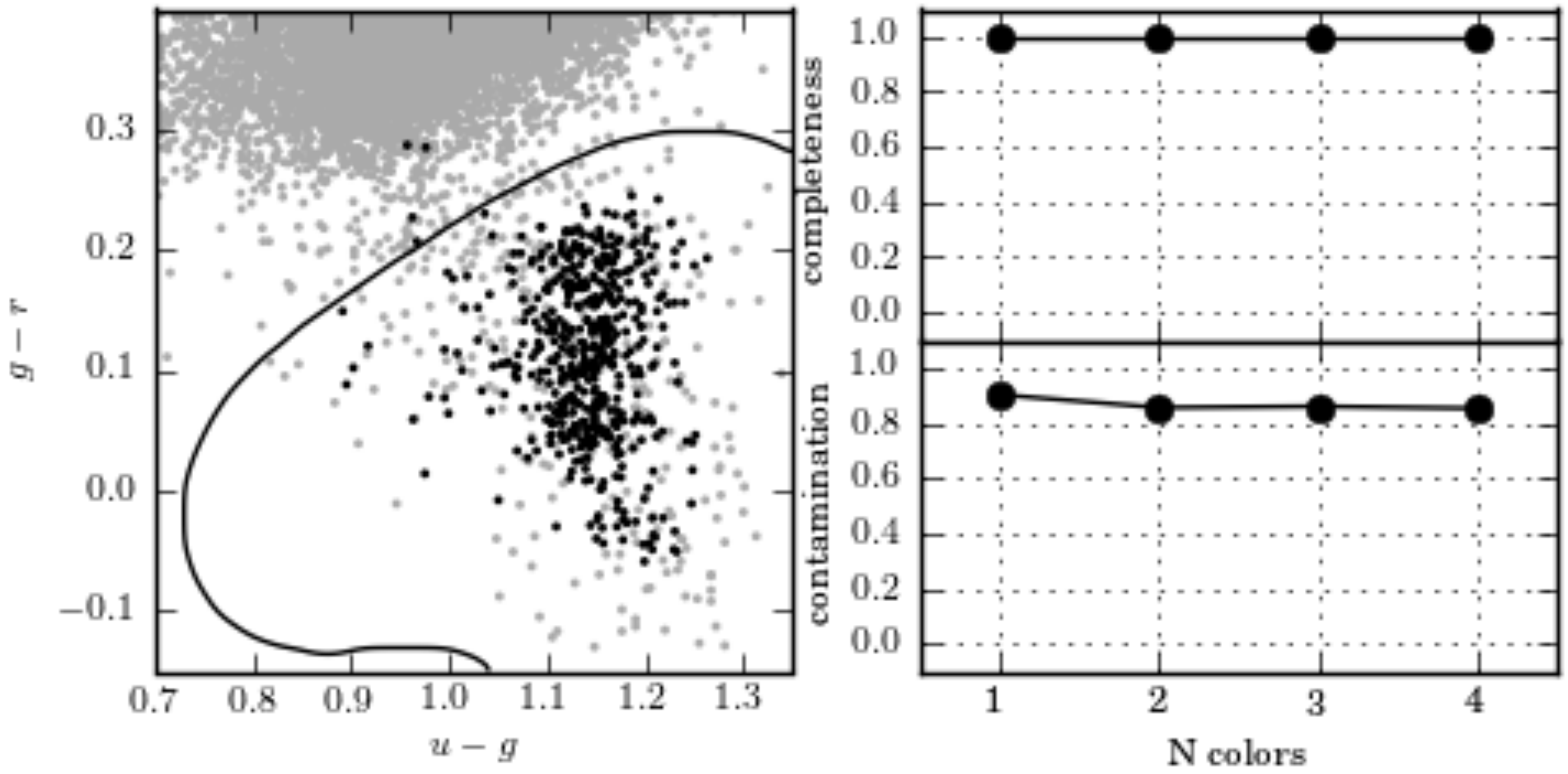


Ivezic et al. (2014)

SVM: kernel trick

- Hyperplane: linear
- Can make boundary non-linear using the *kernel trick*
- Requires the dual representation of the optimization problem for SVM…
- Replace all dot products with K(x,x’) with K a kernel (e.g., Gaussian)

SVM kernel trick example



Ivezic et al. (2014)