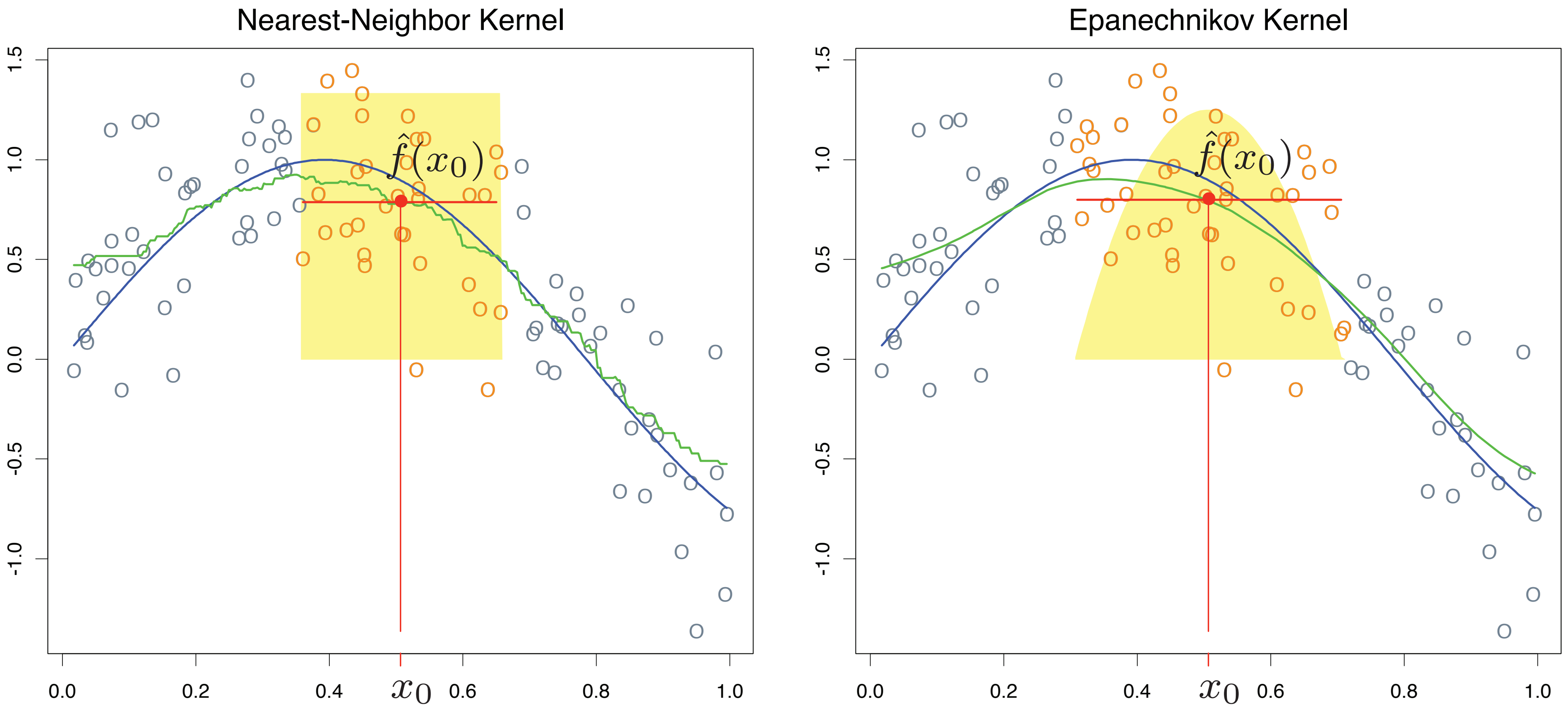
Local regression and kernel methods

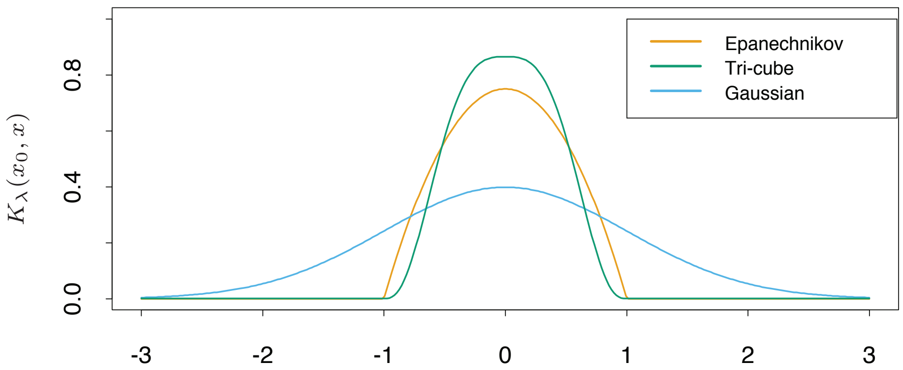
- Local regression
- Varying coefficient models
- Local likelihood
- Kernel density estimation
- Naive Bayes
- Radial basis functions
- Mixture models and EM

$$\hat{f}(x_0) = \text{Ave}(y_i | x_i \in \mathcal{N}_k(x_0))$$


Nearest-Neighbor Kernel

Epanechnikov Kernel

$$\text{Nadaraya-Watson:} \qquad \hat{f}(x_0) = \frac{\sum_{i=1}^{N} K_\lambda(x_0, x_i) y_i}{\sum_{i=1}^{N} K_\lambda(x_0, x_i)}$$



$$K_\lambda(x_0,x) = D\left(\frac{|x-x_0|}{h_\lambda(x_0)}\right)$$

$$h_\lambda(x_0) \quad = \quad \lambda \qquad \sim \text{bias const}$$

$$h_k(x_0) \quad = \quad ||x_0 - x_{[k]}|| \qquad \sim \text{var const}$$

Epanechnikov $D(t) = \begin{cases} \frac{3}{4}(1-t^2) & \text{if } |t| \le 1; \\ 0 & \text{otherwise.} \end{cases}$ Bounded support

Tricube $D(t) = \begin{cases} (1-t^3)^3 & \text{if } |t| \le 1; \\ 0 & \text{otherwise.} \end{cases}$ Bounded support, 2 derivs

Gaussian $D(t) = e^{-\frac{1}{2}t^2}$ Unbounded support, all derivatives

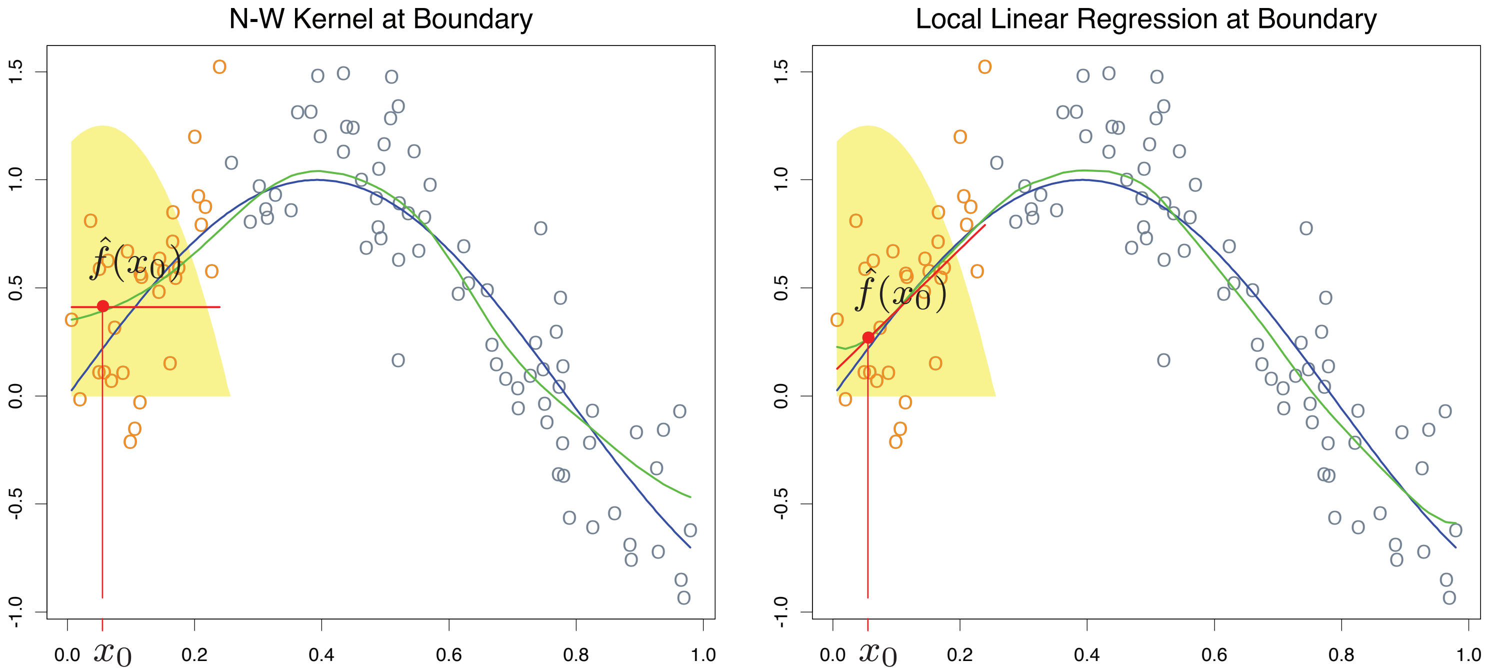
Locally weighted regression

Fit a parametric model locally at x_0 using weighted least squares.
Eg local-linear regression.

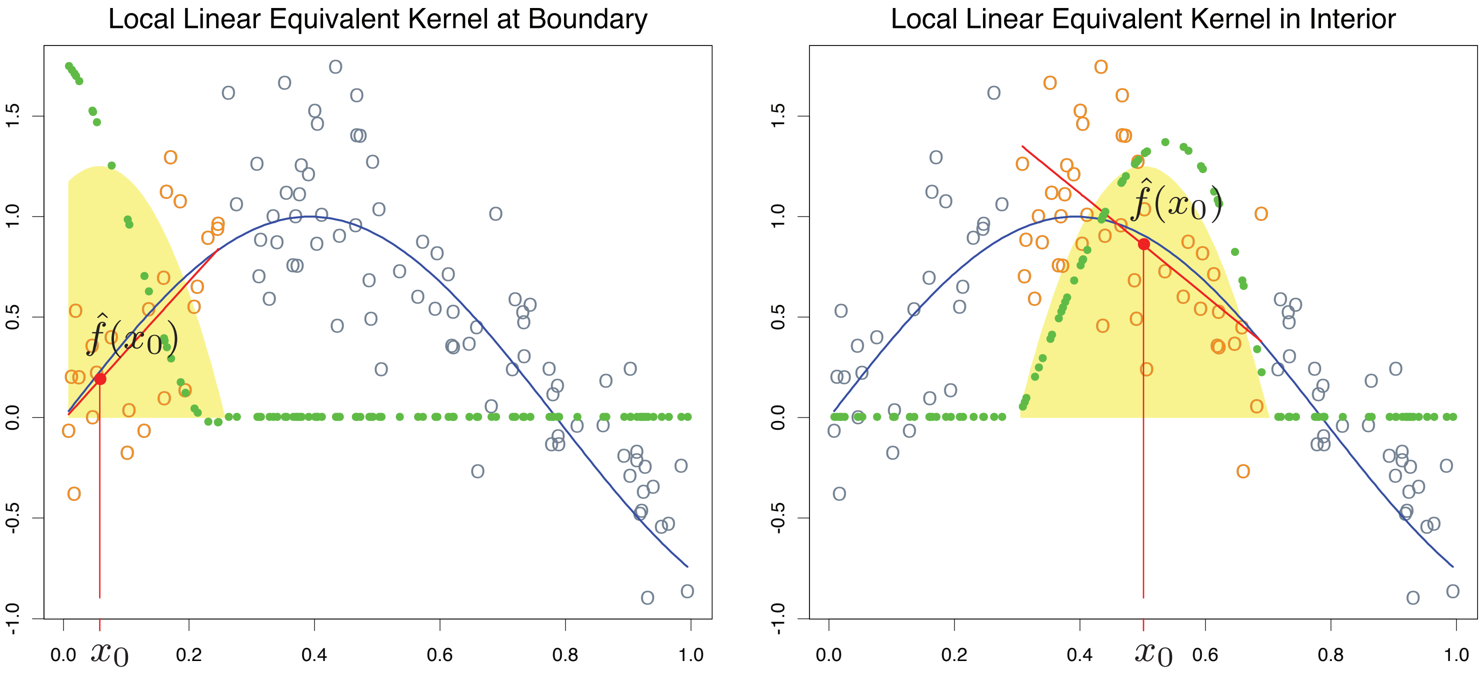
$$\min_{\alpha(x_0),\beta(x_0)} \sum_{i=1}^{N} K_\lambda(x_0, x_i)[y_i - \alpha(x_0) - \beta(x_0)x_i]^2$$

$$\begin{aligned} \hat{f}(x_0) &= \hat{\alpha}(x_0) + \hat{\beta}(x_0)x_0 \\ &= b(x_0)^T(\mathbf{B}^T\mathbf{W}(x_0)\mathbf{B})^{-1}\mathbf{B}^T\mathbf{W}(x_0)\mathbf{y} \\ &= \sum_{i=1}^{N} \ell_i(x_0)y_i \end{aligned}$$

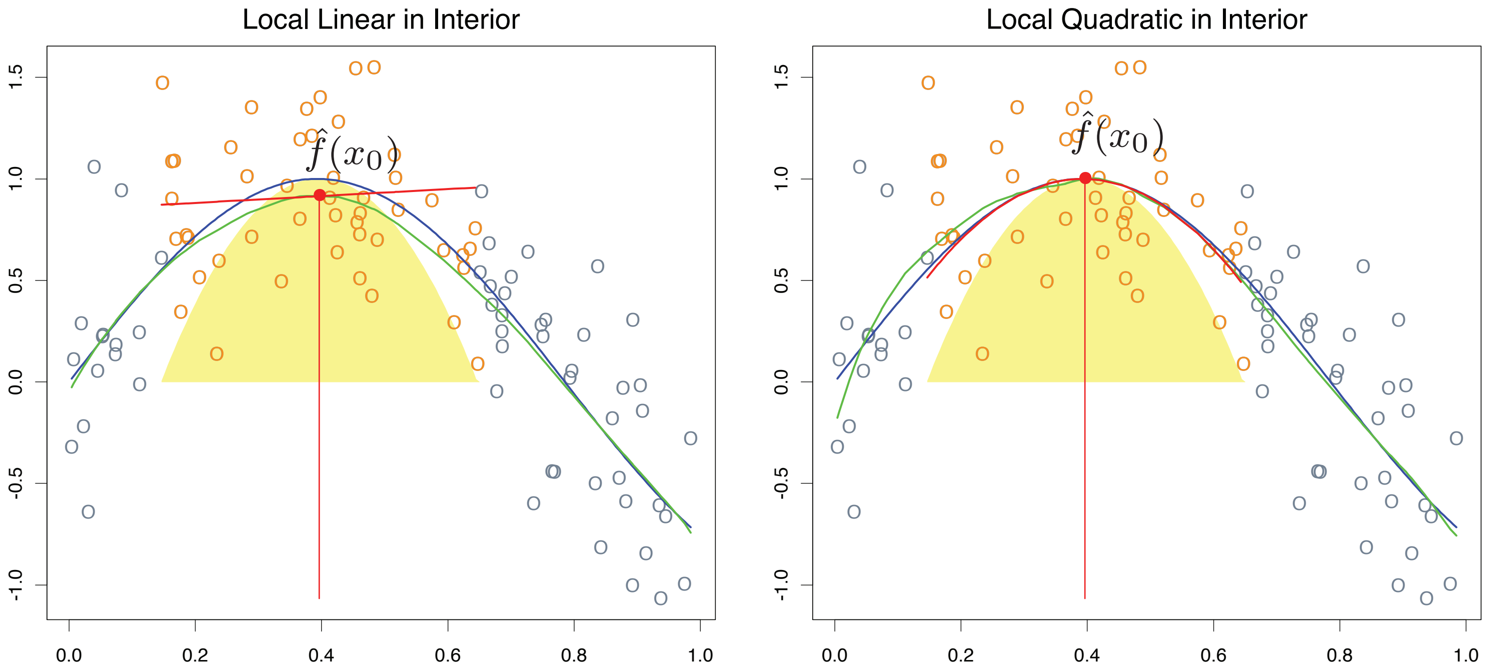
Here $b(x)^T = (1, x)$ and $\mathbf{W}(x_0)$ is a diagonal weight matrix with $W_{ii} = K_\lambda(x_0, x_i)$.



The locally weighted average has bias problems at or near the boundaries of the domain. The true function is approximately linear here, but most of the observations in the neighborhood have a higher mean than the target point, so despite weighting, their mean will be biased upwards. By fitting a locally weighted linear regression (right panel), this bias is removed to first order.



The equivalent kernel $l_i(x_0)$ for local regression. These are the weights in $\hat{f}(x_0) = \sum_{i=1}^{N} l_i(x_0)y_i$, plotted against their corresponding x_i. For display purposes, these have been rescaled, since in fact they sum to 1. The rescaled Nadaraya–Watson kernel is yellow. We see how local regression automatically modifies the local-average weighting kernel to correct for biases due to asymmetry in the smoothing window.



Local linear fits exhibit bias in regions of curvature of the true function.
Local quadratic fits tend to eliminate this bias.

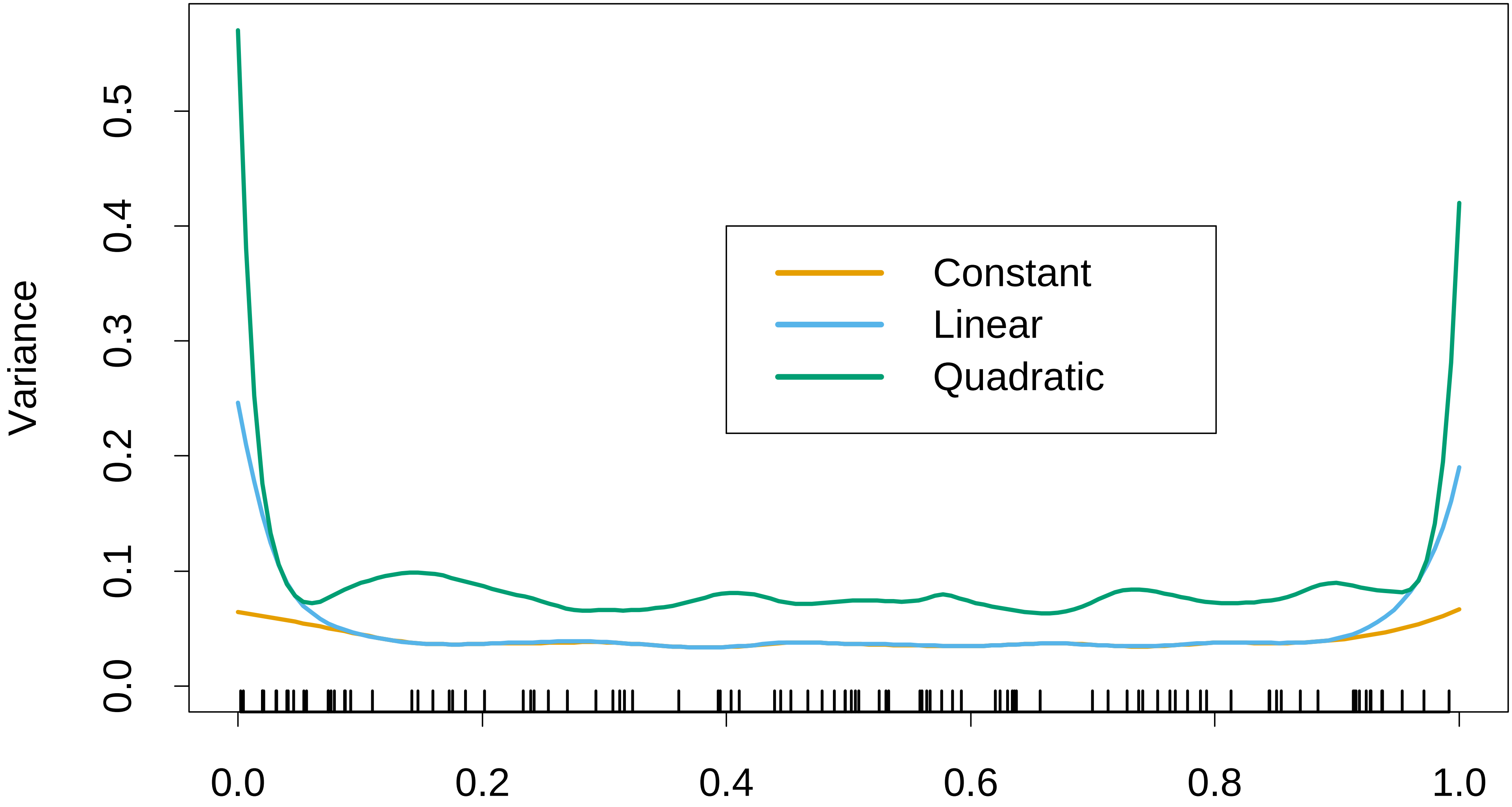
Local polynomial regression and bias

Suppose $y_i = f(x_i) + \varepsilon_i,\ i = 1, \ldots, N$

Lets examine the bias for local-quadratic regression.

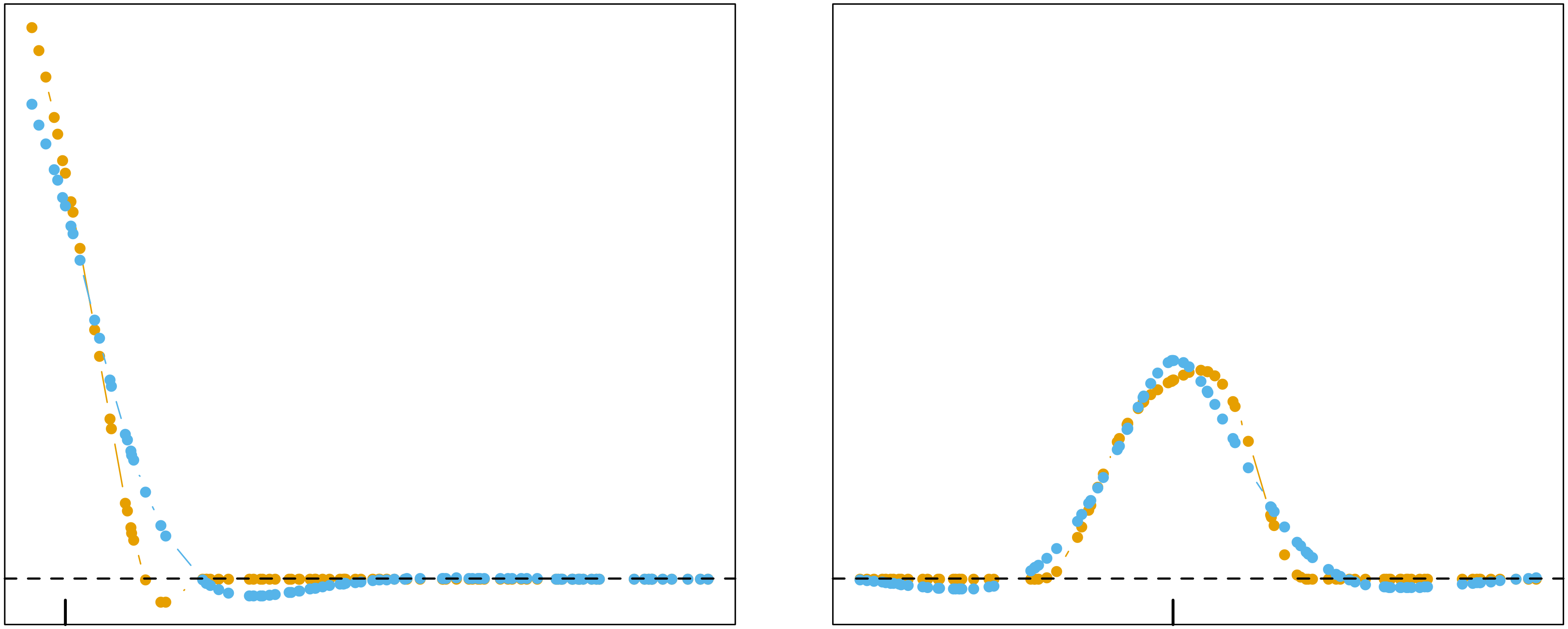
$$\begin{aligned} \mathrm{E}\hat{f}(x_0) &= \sum_{i=1}^{N} l_i(x_0) f(x_i) \\ &= f(x_0)\sum_{i=1}^{N} l_i(x_0) + f'(x_0)\sum_{i=1}^{N}(x_i - x_0) l_i(x_0) \\ &\qquad + \frac{f''(x_0)}{2}\sum_{i=1}^{N}(x_i - x_0)^2 l_i(x_0) + R, \end{aligned}$$

What happens??

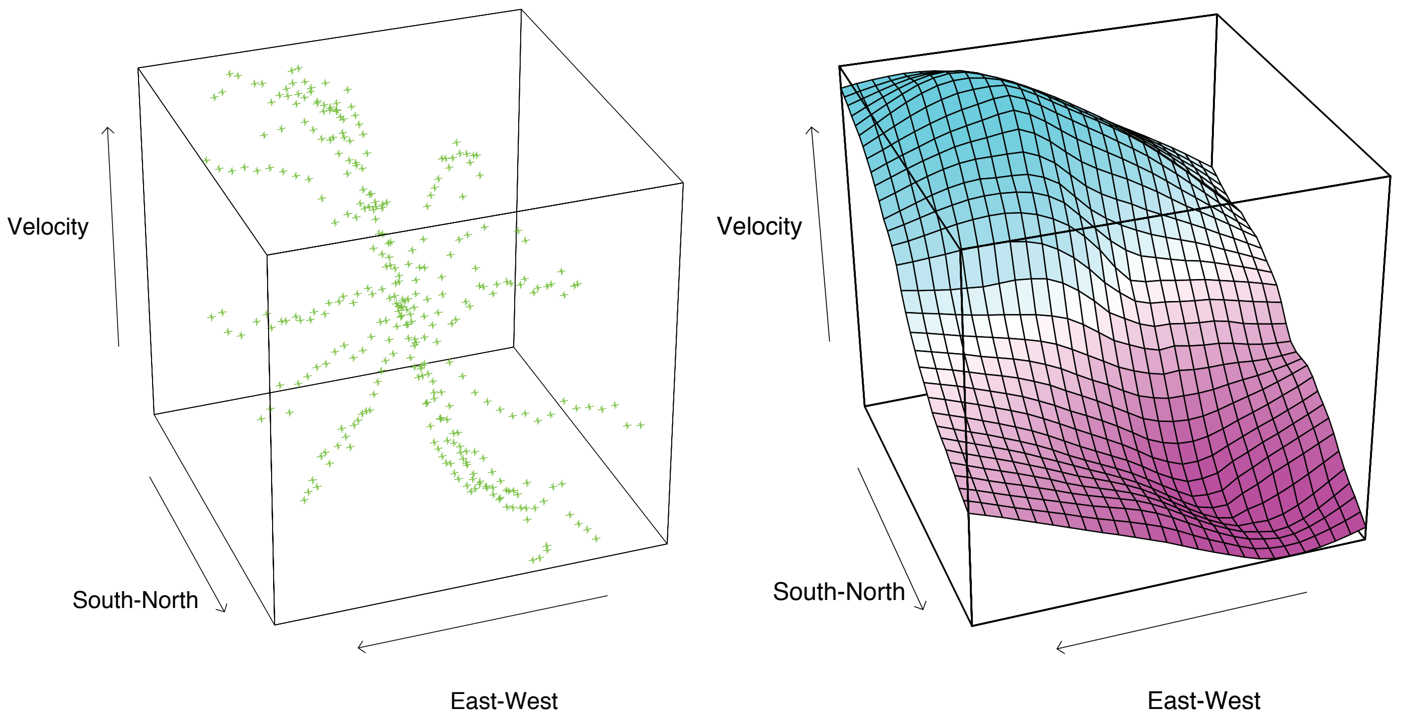


The variances functions $||\ell(x)||^2$ for local constant, linear and quadratic regression, for a metric bandwidth ($\lambda = 0.2$) tri-cube kernel.

Equivalent kernels and Df

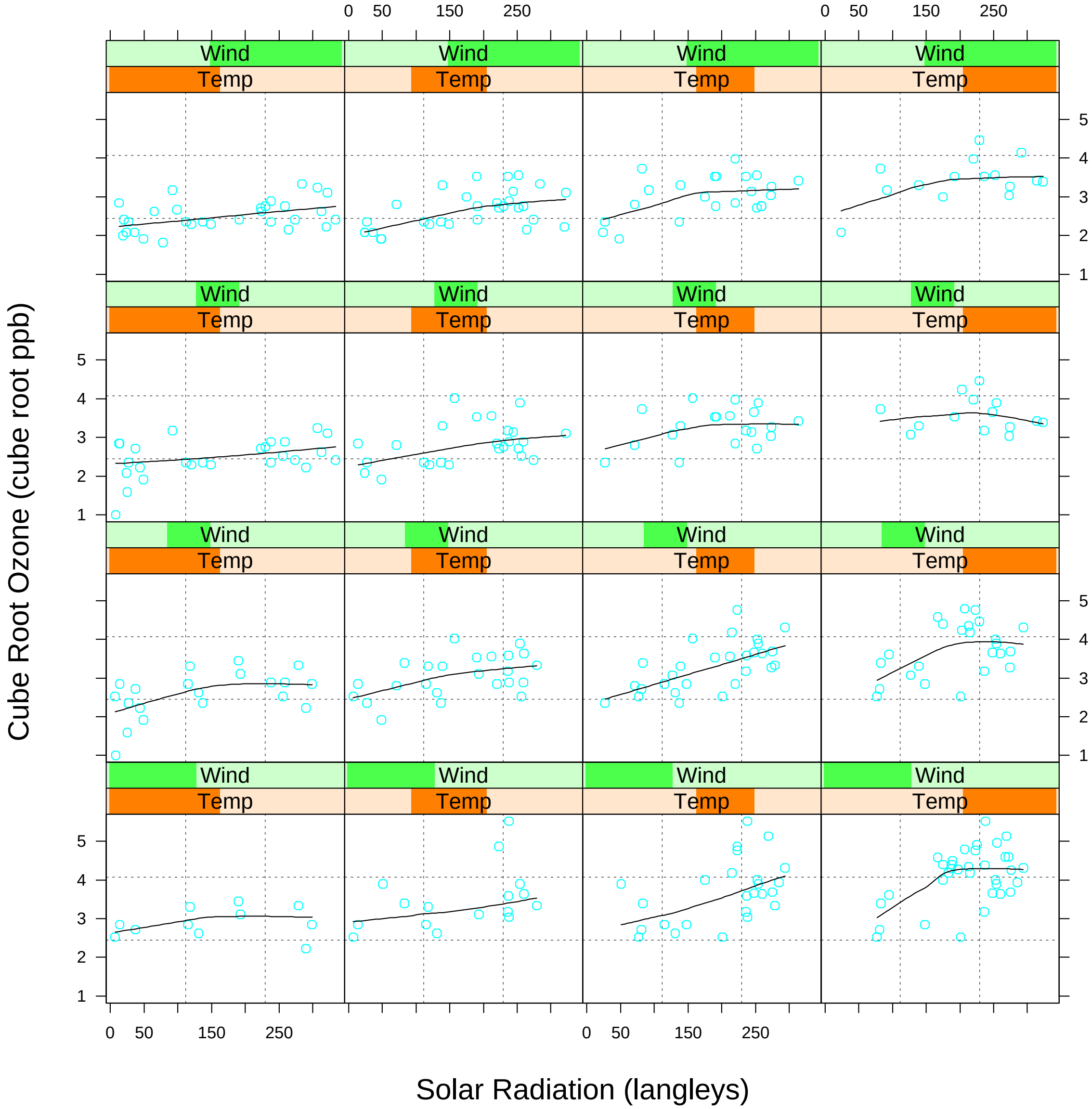


Equivalent kernels for a local linear regression smoother (tri-cube kernel; orange) and a smoothing spline (blue), with matching degrees of freedom: $\mathrm{df}(\lambda) = \mathrm{tr}(\mathbf{S}_\lambda)$. The vertical spikes indicates the target points.



Left: Three-dimensional data, where the response is the velocity measurements on a galaxy, and the two predictors record positions on the celestial sphere. The unusual star-shaped design indicates the way the measurements were made, and results in an extremely irregular boundary. Right: Results of local linear regression smoothing in $\mathbb{R}^2$, using a nearest-neighbor window with 15% of the data.

Conditional plots and the Trellis Display



Varying coefficient models

- Response Y, predictors $X = (X_1, X_2, \ldots, X_p)$, and a special predictor Z (e.g. Time, anything else, or even a collection of variables)
- Assume

$$\begin{aligned} E(Y|X,Z) &= f(X,Z) \\ &= \beta_0(Z) + \beta_1(Z)X_1 + \beta_2(Z)X_2 + \ldots + \beta_q(Z)X_q. \end{aligned}$$

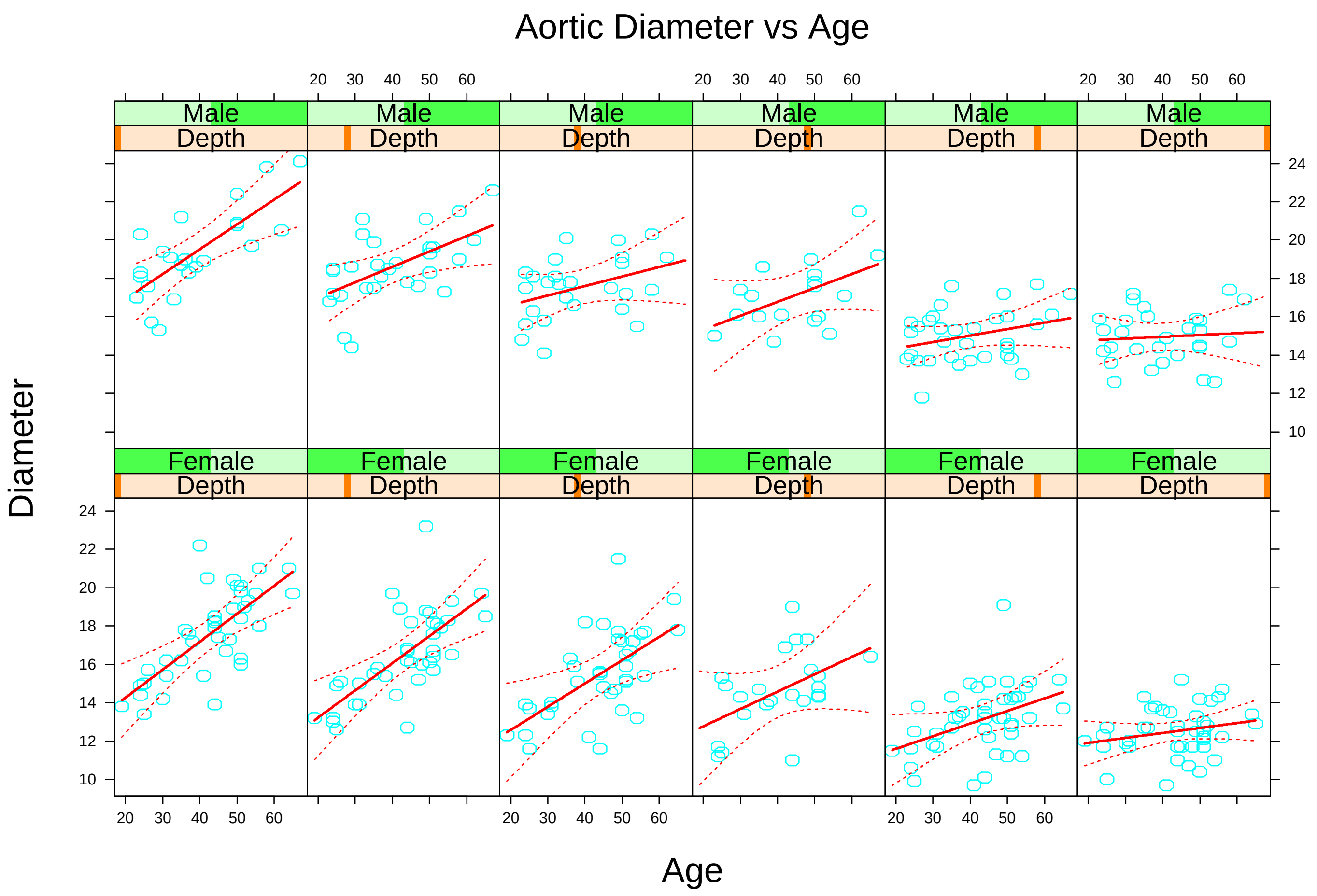
  a.k.a. conditionally linear model or dynamic linear model

This can be viewed as a special form of interaction model.

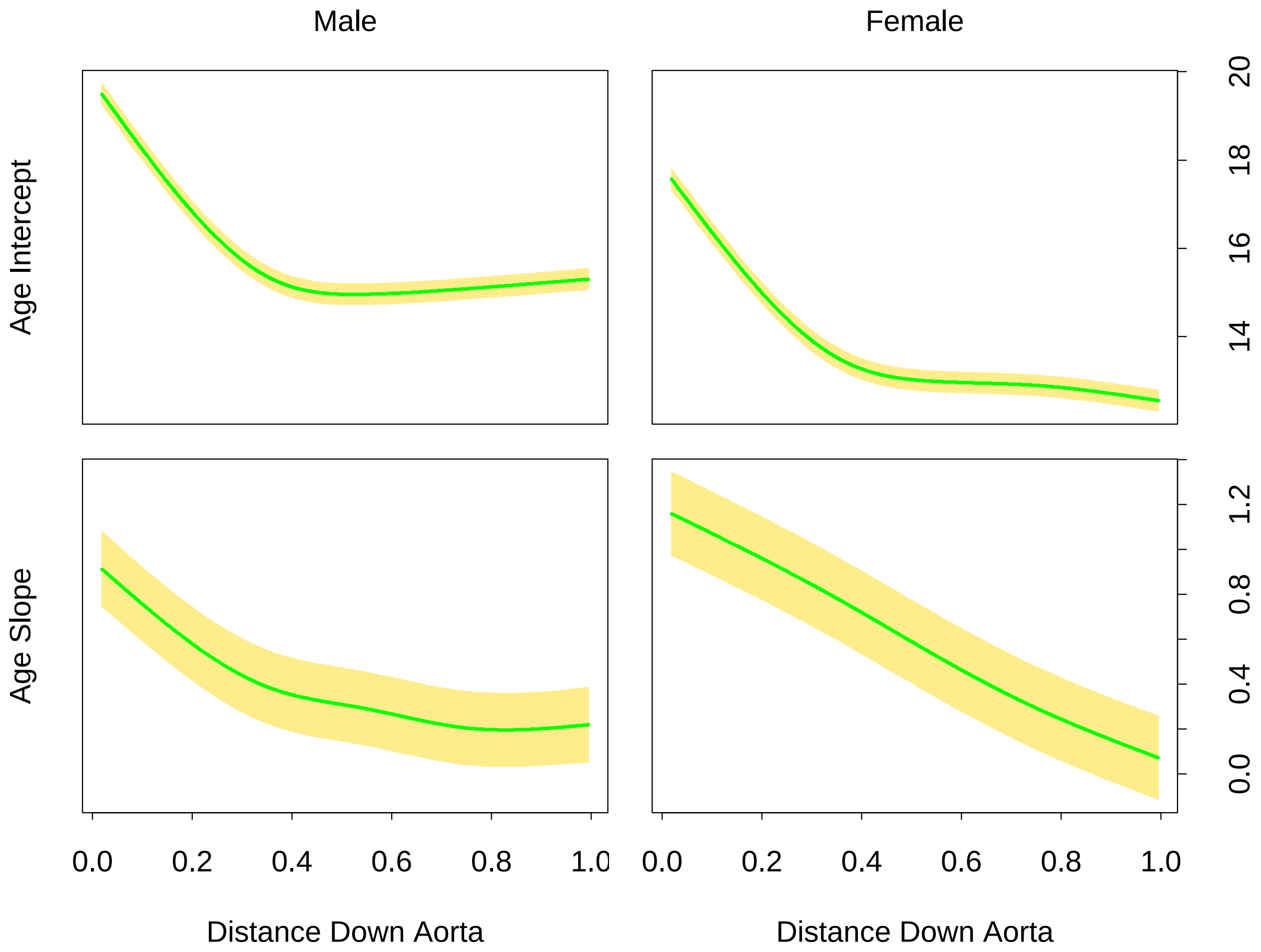
Example

Study of widening of aorta with age. Interested in measuring the rate at which it widens, and to establish if the rate depends on the depth down the aorta where the measurements are made.

$$\begin{aligned} Y &= \text{Diameter of Aorta} \\ X_1 &= \text{Age of patient} \\ Z &= \text{Gender, Depth} \end{aligned}$$



In each panel the `aorta diameter` is modeled as a linear function of `age`. The coefficients vary with `gender` and `depth` down the `aorta` (left is near the top, right is low down).



The intercept and slope of `age` as a function of `distance` down the aorta, separately for males and females. The yellow bands indicate one standard error.

Other Local Models

Local Likelihood

$$\ell(\beta(x_0)) = \sum_{i=1}^{N} K_\lambda(x_0, x_i)\ell(y_i, h(x_i)^T\beta(x_0)), \quad h(x) = (1, x)$$

Local Likelihood VC Models

$$\ell(\beta(z_0)) = \sum_{i=1}^{N} K_\lambda(z_0, z_i)\ell(y_i, x_i^T\beta(z_0))$$

Local AR Models

$$\begin{aligned} y_t &= \beta_0 + \beta_1 y_{t-1} + \ldots + \beta_k y_{t-k} + \epsilon_t,\ t = 0, \ldots, T \\ L_k(t) &= (1, y_{t-1}, \ldots, y_{t-k})^T \\ E(y_t|y_{t-1}, y_{t-2}, \ldots) &= \beta(t)^T L_k(t) \\ \ell(\beta(t_0)) &= \sum_{s=k}^{T} K_\lambda(L_k(t_0), L_k(s))(y_s - L_k(s)^T\beta(t_0))^2 \end{aligned}$$

Example: Logistic Regression

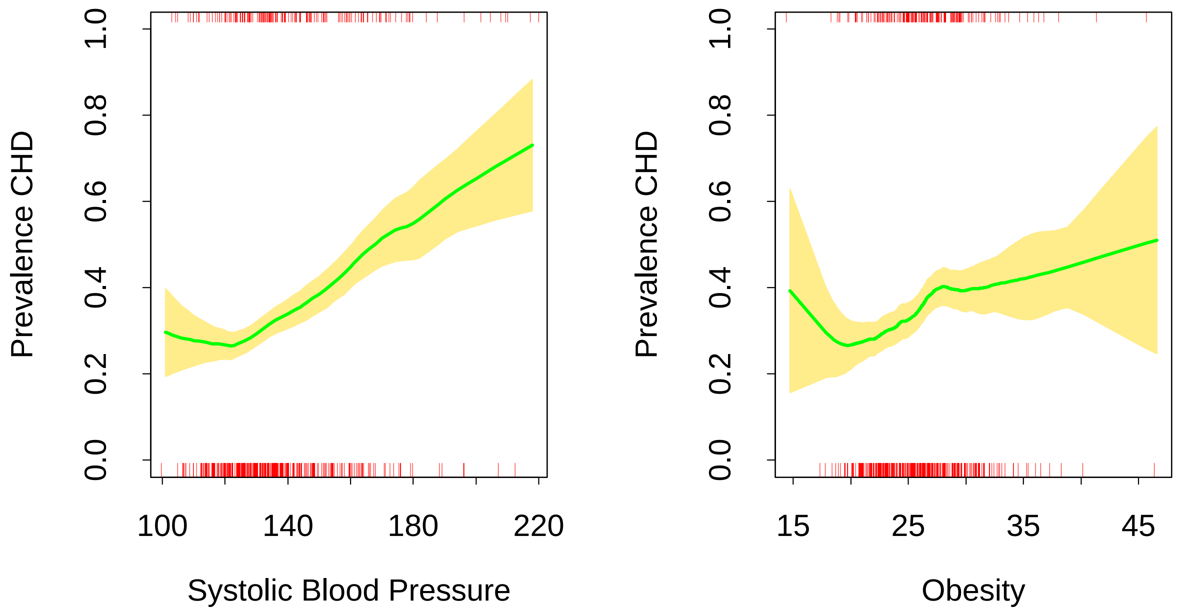
Recall:

$$\Pr(G = j|X = x) = \frac{e^{\beta_{j0}+\beta_j^T x}}{\sum_{k=1}^J e^{\beta_{k0}+\beta_k^T x}} \quad (\beta_{J0} = 0, \beta_J = 0)$$

Local-Likelihood:

$$\sum_{i=1}^N K_\lambda(x_0, x_i)\Bigg\{\beta_{g_i0}(x_0) + \beta_{g_i}(x_0)^T(x_i - x_0) \\ - \log\left[\sum_{k=1}^J \exp\left(\beta_{k0}(x_0) + \beta_k(x_0)^T(x_i - x_0)\right)\right]\Bigg\}.$$

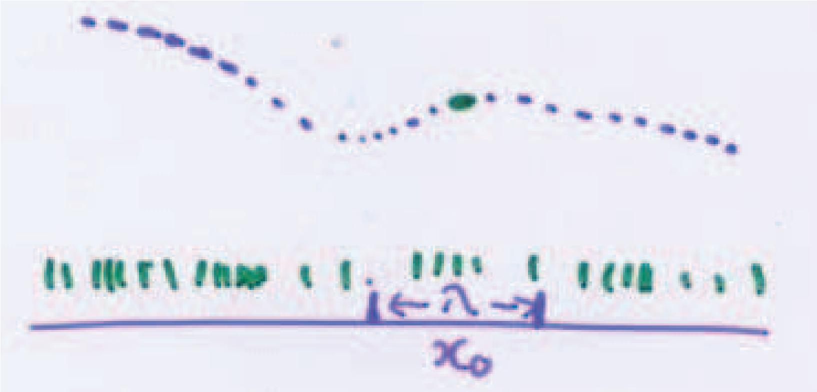
$$\hat{\Pr}(G = j|X = x_0) = \frac{e^{\hat\beta_{j0}(x_0)}}{\sum_{k=1}^J e^{\hat\beta_{k0}(x_0)}}.$$



Each plot shows the binary response CHD as a function of a risk factor for the South African heart disease data. For each plot we have computed the fitted prevalence of CHD using a local linear logistic regression model. The shaded region in the plot indicates an estimated pointwise standard error band.

Kernel Density Estimation

$$\hat{f}_x(x_0) = \frac{\# x_i \in \mathcal{N}(x_0)}{N\lambda}$$

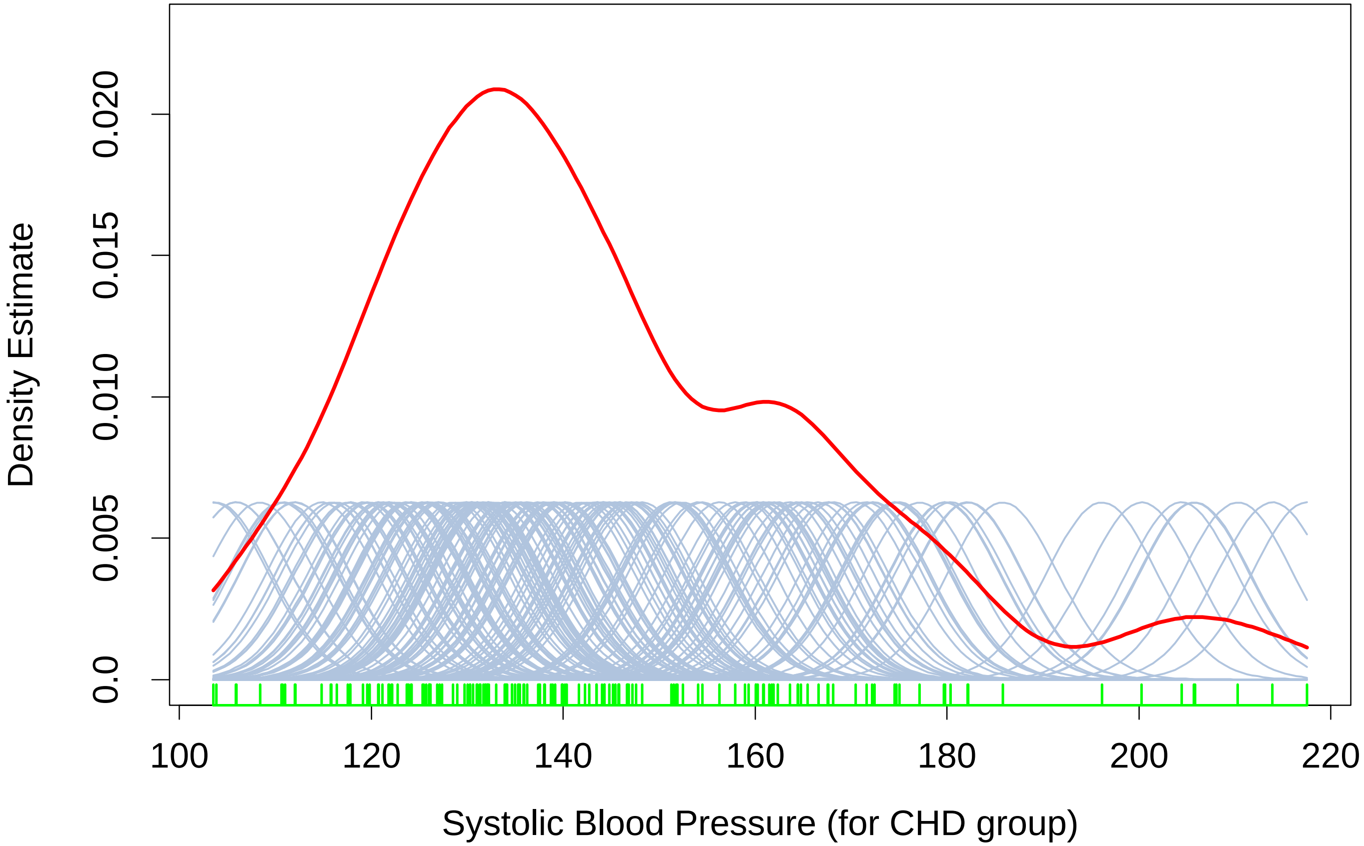


Parzen Window Estimator

$$\hat{f}_x(x_0) = \frac{1}{N\lambda}\sum_{i=1}^{N} K_\lambda(x_0, x_i)$$

E.g. $\frac{1}{\lambda}K_\lambda(x_0, x_i) = \phi_\lambda(x_i - x_0)$ where ϕ_λ is the density of a $N(0, \lambda)$. Note that

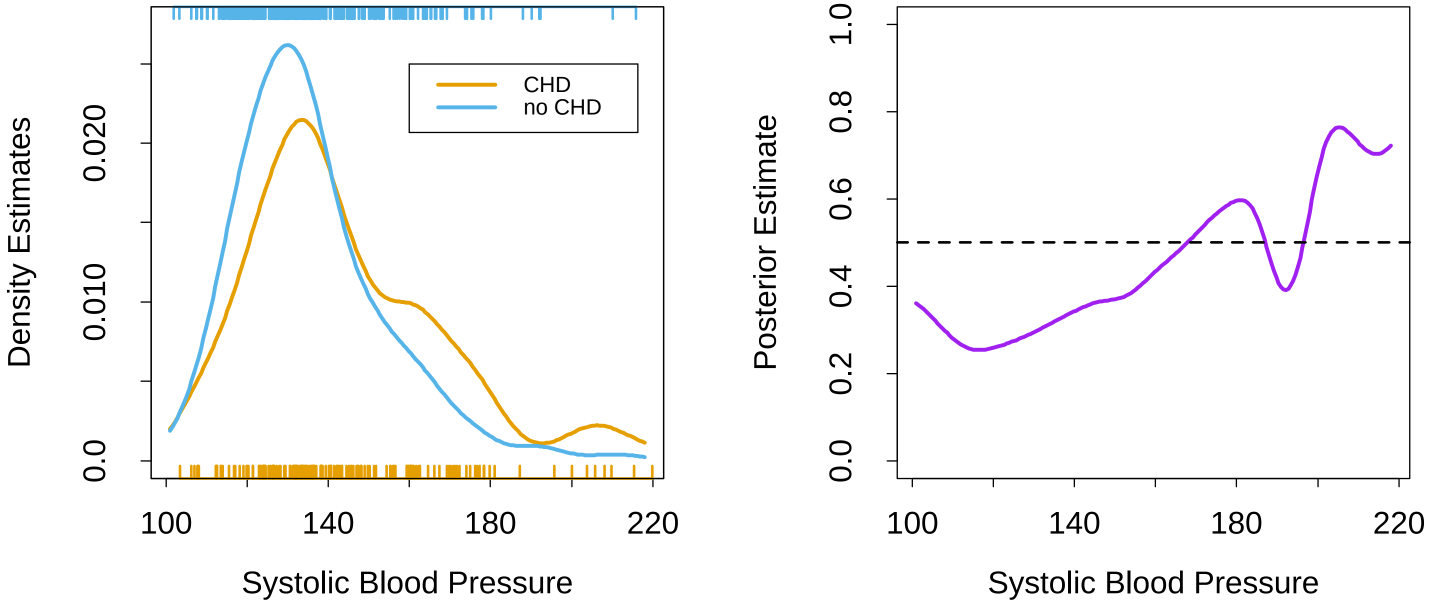
$$\int \hat{f}_x(z)dz = \frac{1}{N}\sum_{i=1}^{N}\int \phi_\lambda(x_i - z)dz = 1$$



A kernel density estimate for systolic blood pressure (for the CHD group). The density estimate at each point is the average contribution from each of the kernels at that point. We have scaled the kernels down by a factor of 10 to make the graph readable.

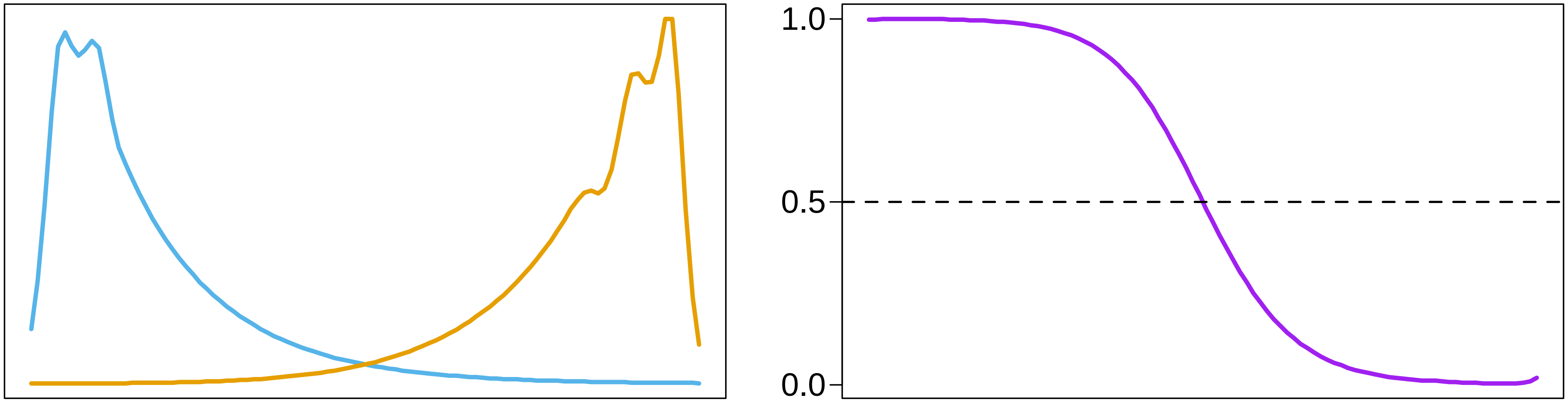
Kernel Density Classification

$$\hat{\Pr}(G = j | X = x_0) = \frac{\hat{\pi}_j \hat{f}_j(x_0)}{\sum_{k=1}^{J} \hat{\pi}_k \hat{f}_k(x_0)}$$



The left panel shows the two separate density estimates for systolic blood pressure in the CHD versus no-CHD groups, using a Gaussian kernel density estimate in each. The right panel shows the estimated posterior probabilities for CHD.

Issues with model selection



The population class densities may have interesting structure (left) that disappears when the posterior probabilities are formed (right).

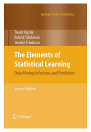
Gaussian mixture models and the EM algorithm

These are similar to kernel density estimates, but with a small number of components (rather than one component per data point)

Outline

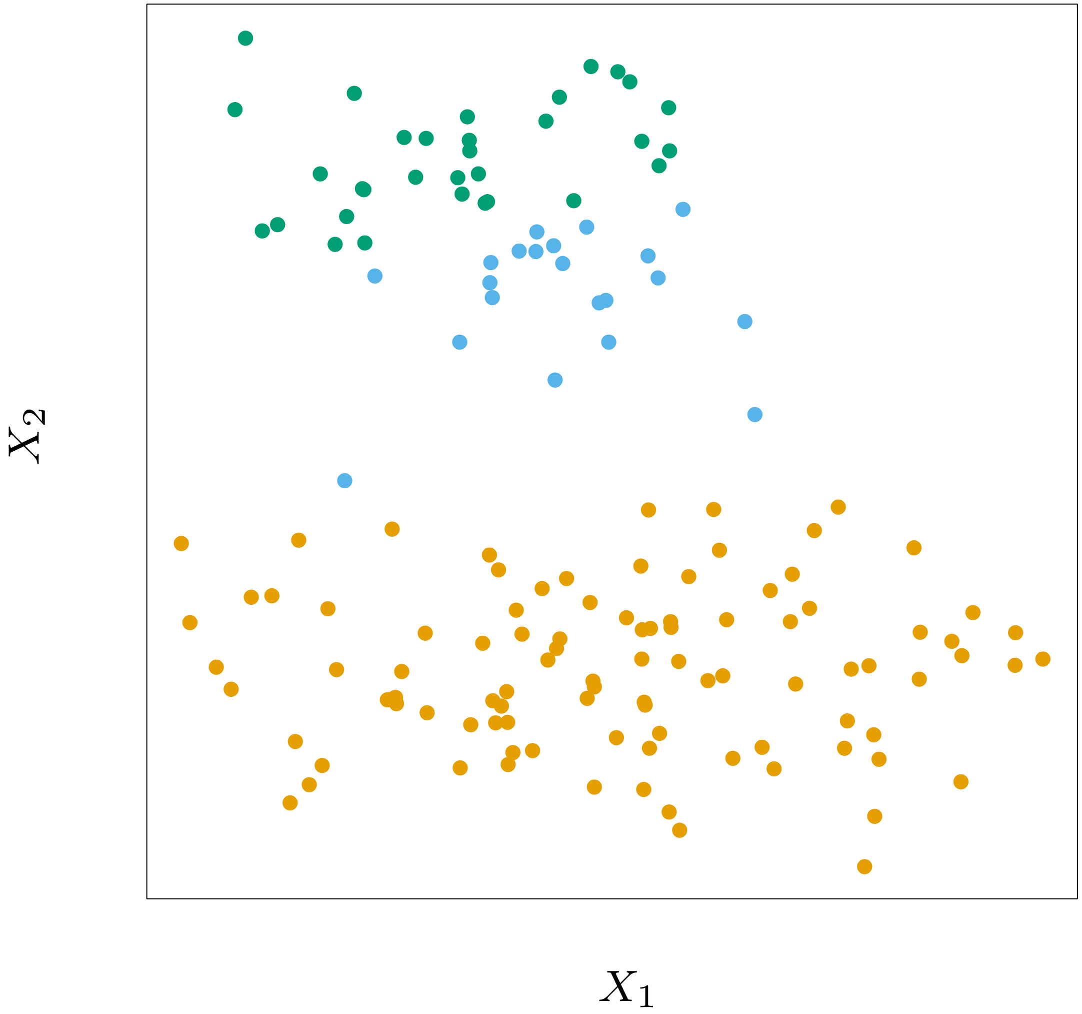
- k-means clustering
- a soft version of k-means: EM algorithm for Gaussian mixture model
- EM algorithm for general missing data problems

K-means clustering

See  pp 460.

Springer Series in Statistics
Trevor Hastie
Robert Tibshirani
Jerome Friedman
The Elements of Statistical Learning
Data Mining, Inference, and Prediction
Second Edition
Springer

Simulated data in the plane, clustered into three classes (represented by orange, blue and green) by the K-means clustering algorithm



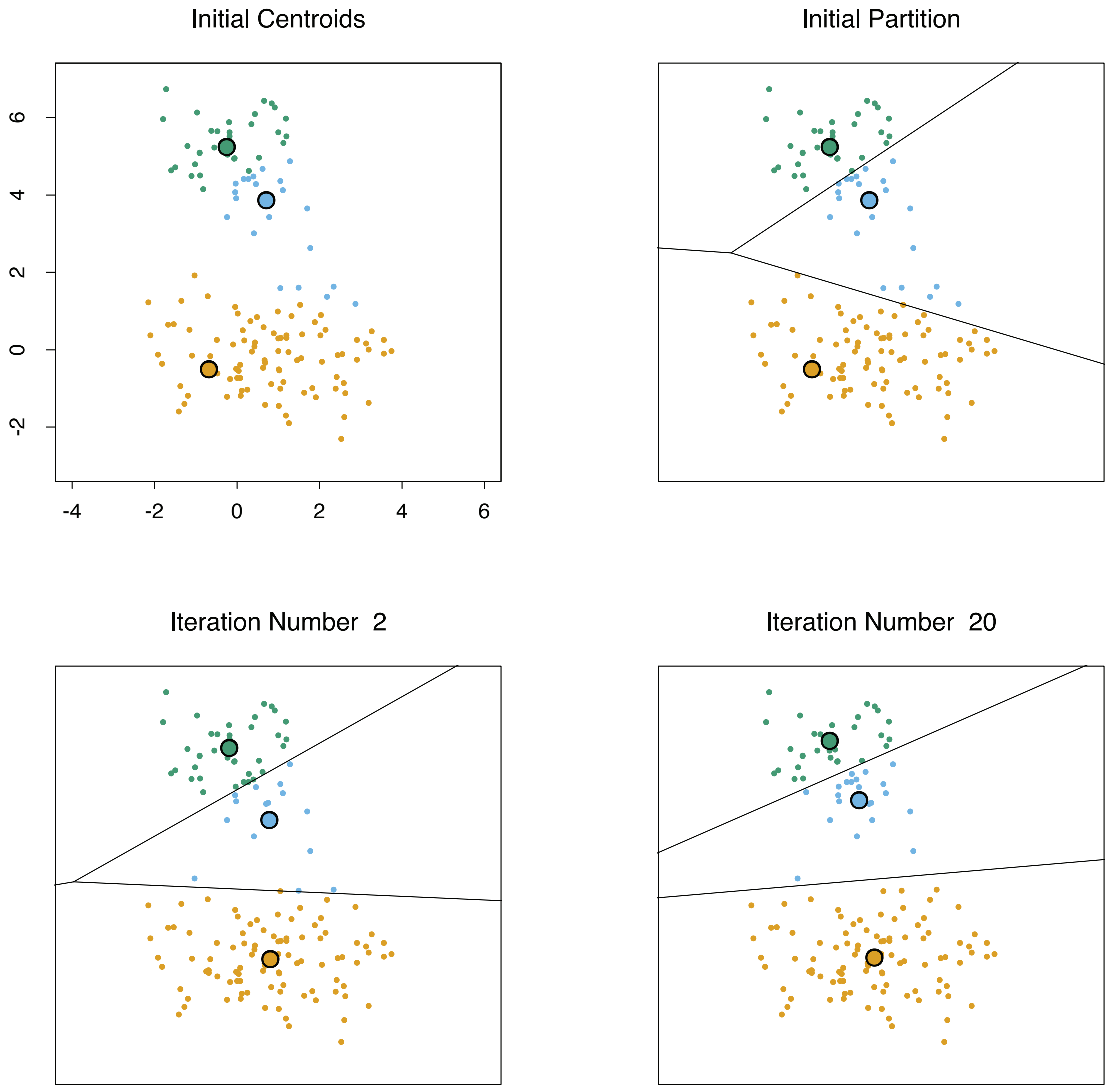
K-means algorithm

1. For each data point, the closest cluster center (in Euclidean distance) is identified;
2. Each cluster center is replaced by the coordinate-wise average of all data points that are closest to it.

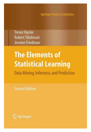
- Steps 1 and 2 are alternated until convergence. Algorithm converges to a local minimum of the within-cluster sum of squares.
- Typically one uses multiple runs from random starting guesses, and chooses the solution with lowest within-cluster sum of squares.

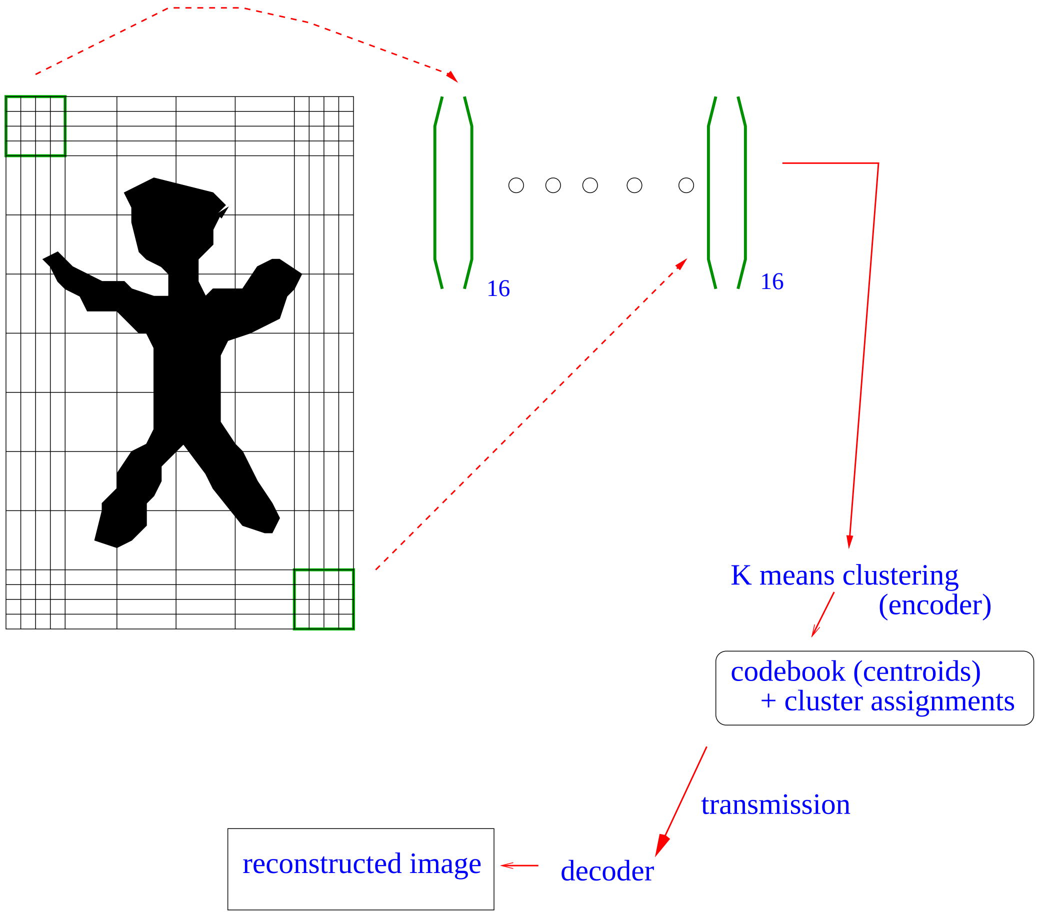
K-means in action

Successive iterations of the K-means clustering algorithm for the simulated data.



Vector Quantization

See  pp 466. VQ is k-means clustering, applied to vectors arising from the blocks of an image (Lloyd algorithm, Gersho & Gray 1991)



K means clustering
(encoder)

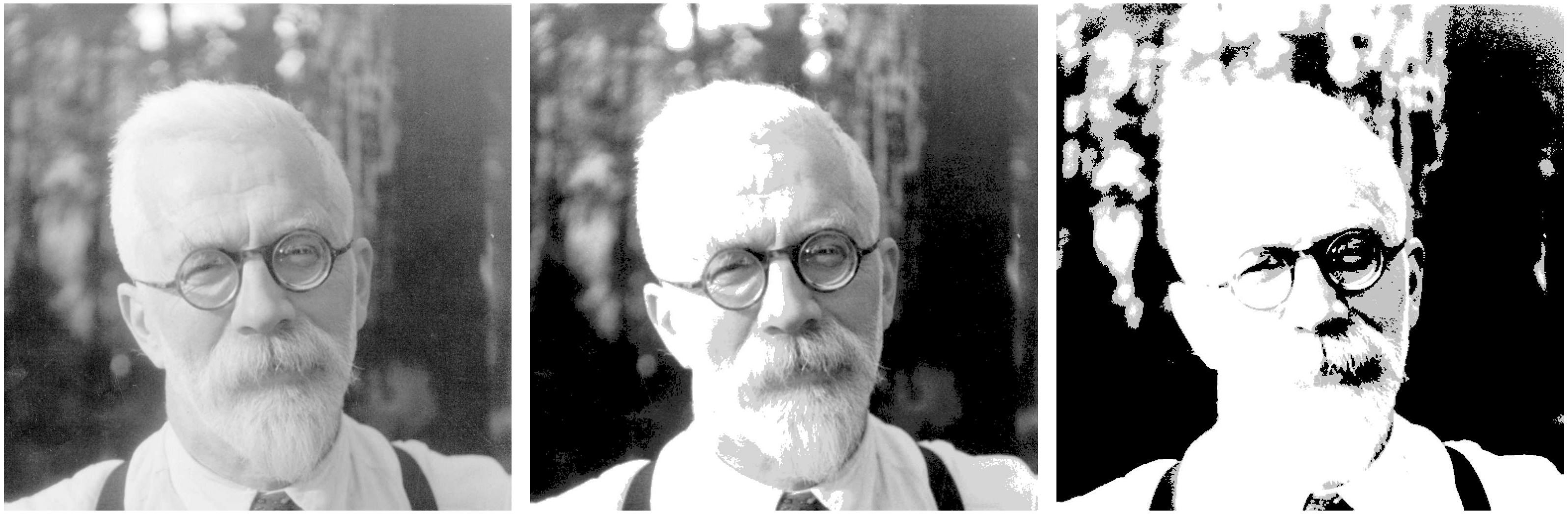
codebook (centroids)
+ cluster assignments

transmission

decoder

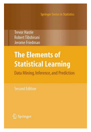
reconstructed image

Real application



Sir Ronald A. Fisher (1890-1962) was one of the founders of modern day statistics, to whom we owe maximum-likelihood, sufficiency, and many other fundamental concepts. The image on the left is a 1024×1024 greyscale image at 8 bits per pixel. The center image is the result of 2×2 block VQ, using 200 code vectors, with a compression rate of 1.9 bits/pixel. The right image uses only four code vectors, with a compression rate of 0.50 bits/pixel.

Mixture Models

See  pp 463. — Soft k-means clustering

Mixture model for density of X:

$$f(X) = \sum_{j=1}^{J} \pi_j g_j(X)$$

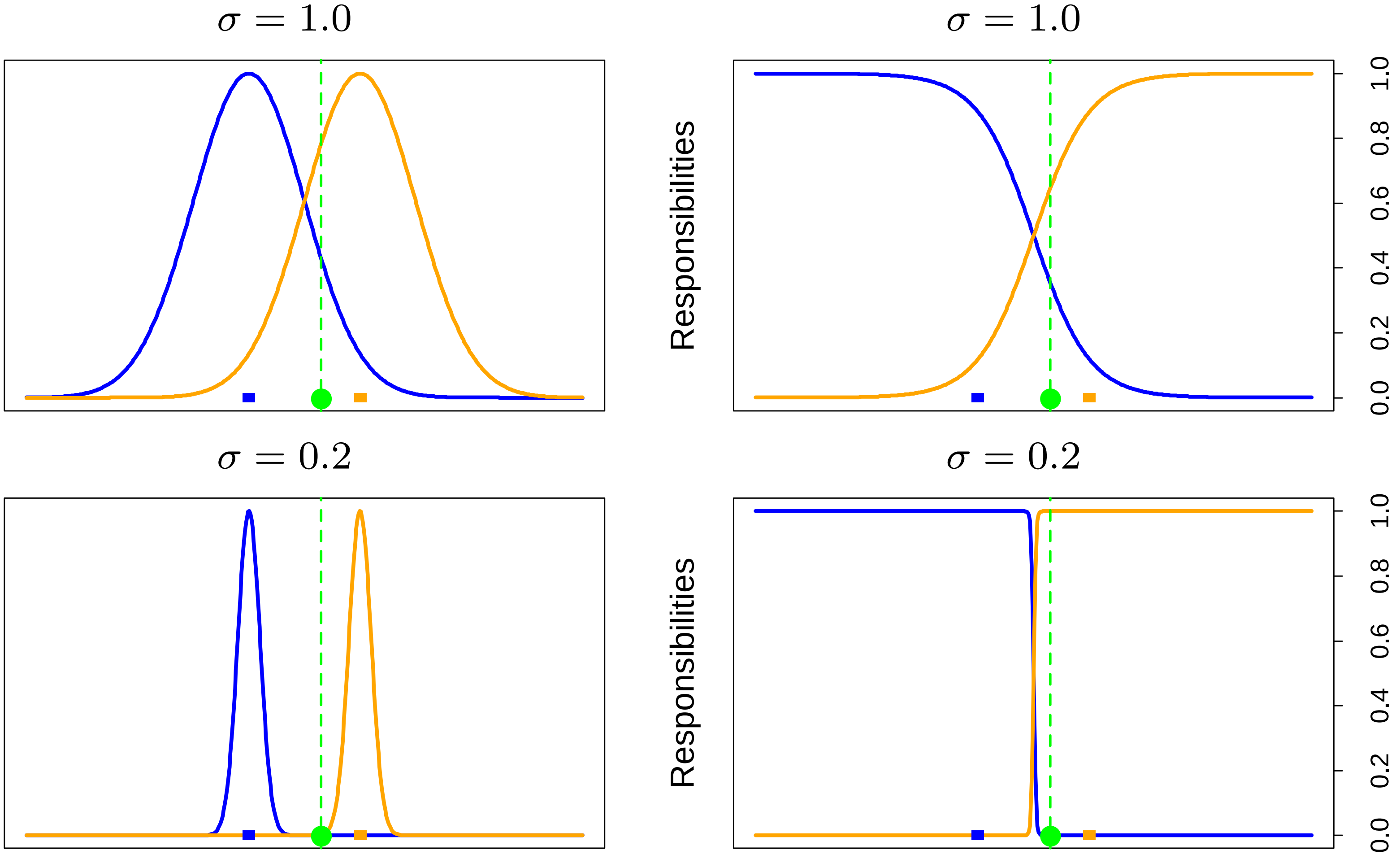
Generative model: to generate an X from f,

1. Generate a "sub-class" $c \in \{1, 2, \ldots, J\}$ at random with probabilities $(\pi_1, \pi_2, \ldots, \pi_J)$.
2. Generate X according to g_c.

Gaussian mixtures: $g_j(X) = \phi(X; \mu_j, \Sigma_j)$.

Often we assume $\Sigma_j = \sigma_j^2 \cdot I$.

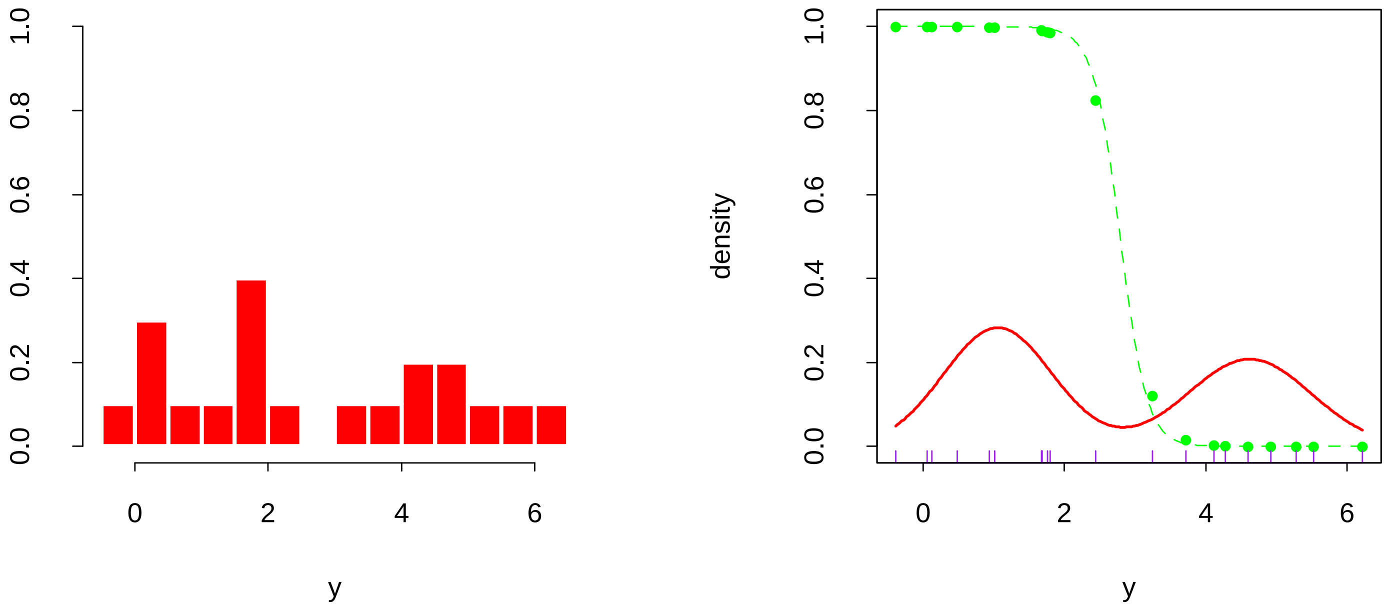
Soft assignment of “responsibilities”



Details of figure

- Left panels: two Gaussian densities $g_1(x)$ and $g_2(x)$ (blue and orange) on the real line, and a single data point (green dot) at $x = 0.5$. The colored squares are plotted at $x = -1.0$ and $x = 1.0$, the means of each density.
- Right panels: the relative densities $g_1(x)/(g_1(x) + g_2(x))$ and $g_2(x)/(g_1(x) + g_2(x))$, called the "responsibilities" of each cluster, for this data point. In the top panels, the Gaussian standard deviation $\sigma = 1.0$; in the bottom panels $\sigma = 0.2$.
- The EM algorithm uses these responsibilities to make a "soft" assignment of each data point to each of the two clusters. When σ is fairly large, the responsibilities can be near 0.5 (they are 0.36 and 0.64 in the top right panel).
- As $\sigma \rightarrow 0$, the responsibilities $\rightarrow 1$, for the cluster center closest to the target point, and 0 for all other clusters. This "hard" assignment is seen in the bottom right panel.

The EM Algorithm



Two-Component Mixture Model. Left panel: histogram of 20 fictitious data points. Right panel: maximum likelihood fit of Gaussian mixture model (solid red) and responsibility (dotted green) of the left component density for observation y, as a function of y.

-0.39	0.12	0.94	1.67	1.76	2.44	3.72	4.28	4.92	5.53
0.06	0.48	1.01	1.68	1.80	3.25	4.12	4.60	5.28	6.22

$$\begin{aligned} Y_1 &\sim N(\mu_1, \sigma_1^2), \\ Y_2 &\sim N(\mu_2, \sigma_2^2), \\ Y &= (1-\Delta)\cdot Y_1 + \Delta \cdot Y_2, \end{aligned}$$

where $\Delta \in \{0, 1\}$ with $\Pr(\Delta = 1) = \pi$.

Let $\phi_\theta(x)$ denote the normal density with parameters $\theta = (\mu, \sigma^2)$. Then the density of Y is

$$g_Y(y) = \pi\phi_{\theta_1}(y) + (1-\pi)\phi_{\theta_2}(y).$$

Collect all the parameters in $\Omega = \{\pi, \theta_1, \theta_2\}$.

The log-likelihood based on the N training cases is

$$\ell(\Omega; \mathbf{y}) = \sum_{i=1}^{N} \log[\pi\phi_{\theta_1}(y_i) + (1-\pi)\phi_{\theta_2}(y_i)].$$

Direct maximization of $\ell(\Omega; \mathbf{y})$ is quite difficult numerically, because of the sum of terms inside the logarithm. There is, however, a simpler approach. We consider unobserved latent variables Δ_i taking values 0 or 1: if $\Delta_i = 1$ then Y_i comes from model 1, otherwise it comes from model 2.

Suppose we knew the values of the Δ_i's. Then the log-likelihood of $(\mathbf{y}, \boldsymbol{\Delta})$ would be

$$\begin{aligned}\ell_0(\Omega; \mathbf{y}, \boldsymbol{\Delta}) &= \sum_{i=1}^{N} [\Delta_i \log \phi_{\theta_1}(y_i) + (1-\Delta_i) \log \phi_{\theta_2}(y_i)] \\ &\quad + \sum_{i=1}^{N} [\Delta_i \log \pi + (1-\Delta_i) \log(1-\pi)]\end{aligned}$$

Since the values of the Δ_i's are actually unknown, we proceed in an iterative fashion, substituting for each Δ_i its expected value

$$\gamma_i(\Omega) = \mathrm{E}\left(\Delta_i|\Omega, \mathbf{y}\right) = \Pr(\Delta_i = 1|\Omega, \mathbf{y}),$$

also called the *responsibility* of model 1 for observation i. We use a procedure called the EM algorithm.

EM algorithm for two-component Gaussian mixture

- Take initial guesses for the parameters $\hat{\mu}_1, \hat{\sigma}_1^2, \hat{\mu}_2, \hat{\sigma}_2^2, \hat{\pi}$ (see text).
- *Expectation Step*: compute the responsibilities

$$\hat{\gamma}_i = \frac{\hat{\pi}\phi_{\hat{\theta}_1}(y_i)}{\hat{\pi}\phi_{\hat{\theta}_1}(y_i) + (1-\hat{\pi})\phi_{\hat{\theta}_2}(y_i)},\ i = 1, 2, \ldots, N.$$

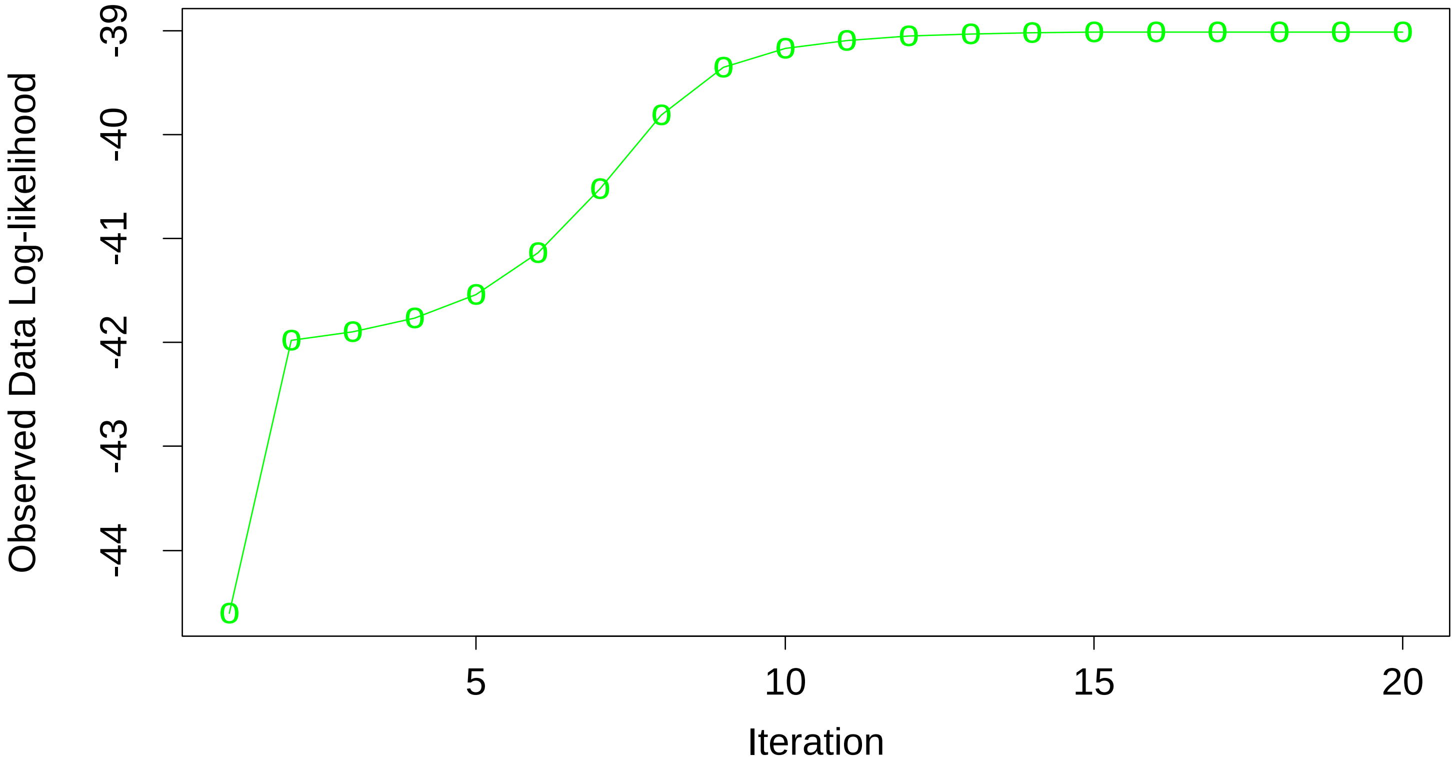
- *Maximization Step*: compute the weighted means and variances:

$$\hat{\mu}_1 = \frac{\sum_{i=1}^N \hat{\gamma}_i y_i}{\sum_{i=1}^N \hat{\gamma}_i}, \qquad \hat{\sigma}_1^2 = \frac{\sum_{i=1}^N \hat{\gamma}_i (y_i - \hat{\mu}_1)^2}{\sum_{i=1}^N \hat{\gamma}_i},$$

$$\hat{\mu}_2 = \frac{\sum_{i=1}^N (1-\hat{\gamma}_i) y_i}{\sum_{i=1}^N (1-\hat{\gamma}_i)}, \qquad \hat{\sigma}_2^2 = \frac{\sum_{i=1}^N (1-\hat{\gamma}_i)(y_i - \hat{\mu}_2)^2}{\sum_{i=1}^N (1-\hat{\gamma}_i)},$$

  and the mixing probability $\hat{\pi} = \sum_{i=1}^N \hat{\gamma}_i / N$.

- Iterate these steps until convergence.



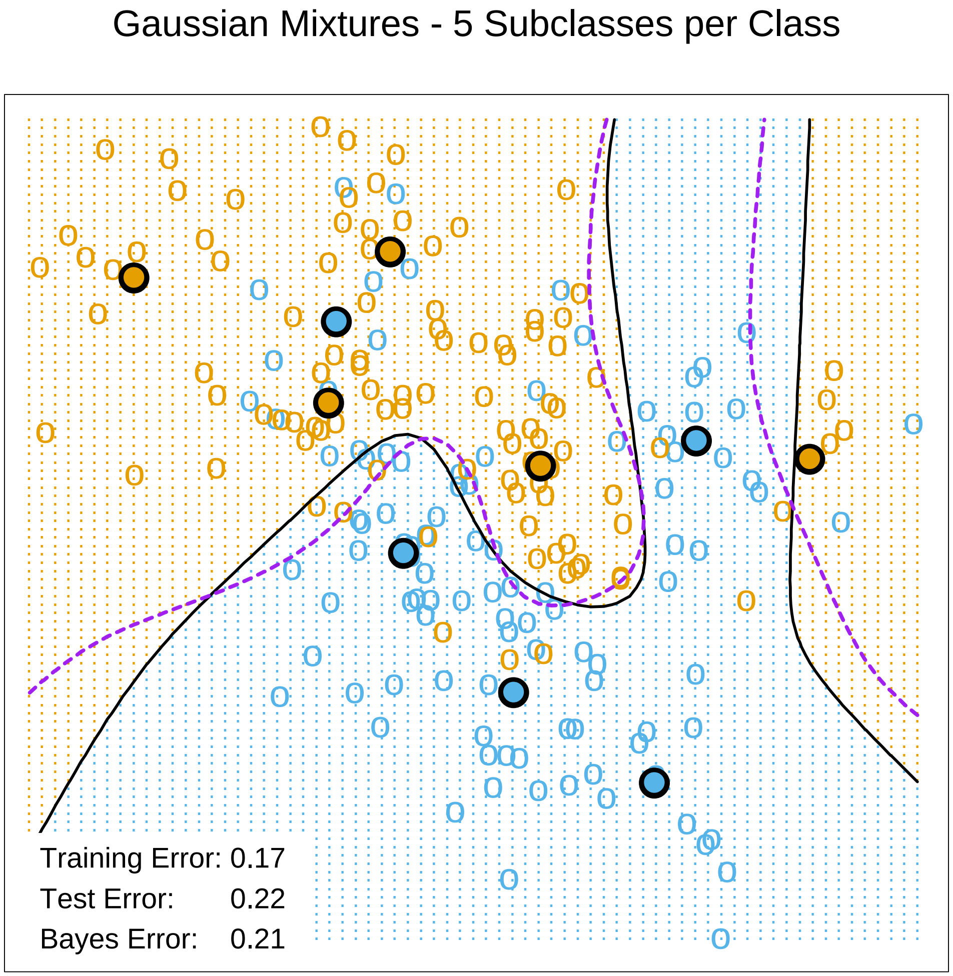
EM algorithm: observed data log-likelihood as a function of the iteration number

Selected iterations of the EM algorithm for mixture example.

Iteration	$\hat{\pi}$
1	0.485
5	0.493
10	0.523
15	0.544
20	0.546

The final maximum likelihood estimates are

$$
\begin{aligned}
\hat{\mu}_1 &= 4.62, & \hat{\sigma}_1^2 &= 0.87, \\
\hat{\mu}_2 &= 1.06, & \hat{\sigma}_2^2 &= 0.77, \\
& & \hat{\pi} = 0.546. &
\end{aligned}
$$



Gaussian mixture model used to classify the mixture data.

EM for general missing data problems

Dempster, Laird and Rubin (1977)

- Our observed data is $\mathbf{z}$, having log-likelihood $\ell(\Omega; \mathbf{z})$ depending on parameters Ω.
- The latent or missing data is $\mathbf{z}^m$, so that the complete data is $\mathbf{t} = (\mathbf{z}, \mathbf{z}^m)$ with log-likelihood $\ell_0(\Omega; \mathbf{t})$, and ℓ_0 based on the complete-data density.
- In the mixture problem $(\mathbf{z}, \mathbf{z}^m) = (\mathbf{y}, \Delta)$.
- EM paper in 1977 has interesting discussion— many (including Hartley and Baum) said that they had already done this work!

The EM algorithm.

1. Start with initial guesses for the parameters $\hat{\Omega}^{(0)}$.
2. *Expectation Step*: at the jth step, compute

   $$Q(\Omega', \hat{\Omega}^{(j)}) = \mathrm{E}\left(\ell_0(\Omega'; \mathbf{t}) | \mathbf{z}, \hat{\Omega}^{(j)}\right)$$

   as a function of the dummy argument Ω'.
3. *Maximization Step*: determine the new estimate $\hat{\Omega}^{(j+1)}$ as the maximizer of $Q(\Omega', \hat{\Omega}^{(j)})$ over Ω'.
4. Iterate steps 2 and 3 until convergence.

Proof that EM works

Since

$$\Pr(\mathbf{z}^m|\mathbf{z},\Omega') = \frac{\Pr(\mathbf{z}^m,\mathbf{z}|\Omega')}{\Pr(\mathbf{z}|\Omega')},$$

we can write

$$\Pr(\mathbf{z}|\Omega') = \frac{\Pr(\mathbf{t}|\Omega')}{\Pr(\mathbf{z}^m|\mathbf{z},\Omega')}.$$

In terms of log-likelihoods, we have $\ell(\Omega';\mathbf{z}) = \ell_0(\Omega';\mathbf{t}) - \ell_1(\Omega';\mathbf{z}^m|\mathbf{z})$, where ℓ_1 is based on the conditional density $\Pr(\mathbf{z}^m|\mathbf{z},\Omega')$. Taking conditional expectations with respect to the distribution of $\mathbf{t}|\mathbf{z}$ governed by parameter Ω gives

$$\begin{aligned} \ell(\Omega';\mathbf{z}) &= \mathrm{E}\left[\ell_0(\Omega';\mathbf{t})|\mathbf{z},\Omega\right] - \mathrm{E}\left[\ell_1(\Omega';\mathbf{z}^m|\mathbf{z})|\mathbf{z},\Omega\right] \\ &\equiv Q(\Omega',\Omega) - R(\Omega',\Omega). \end{aligned}$$

In the M step, the EM algorithm maximizes $Q(\Omega', \Omega)$ over Ω', rather than the actual objective function $\ell(\Omega'; \mathbf{z})$.

Why does it succeed in maximizing $\ell(\Omega'; \mathbf{z})$? Note that $R(\Omega^*, \Omega)$ is the expectation of a log-likelihood of a density (indexed by Ω^*), with respect to the same density indexed by Ω, and hence (by Jensen's inequality) is maximized as a function of Ω^*, when $\Omega^* = \Omega$ (see Exercise 8.1). So if Ω' maximizes $Q(\Omega', \Omega)$, we see that

$$\begin{aligned} \ell(\Omega'; \mathbf{z}) - \ell(\Omega; \mathbf{z}) &= [Q(\Omega', \Omega) - Q(\Omega, \Omega)] - [R(\Omega', \Omega) - R(\Omega, \Omega)] \\ &\geq 0. \end{aligned}$$

Hence the M step never decreases the log-likelihood.