Dimension Reduction

David Hofmeyr

Dept. Statistics and Actuarial Science,
Stellenbosch University, South Africa

6-8 December 2021

forward together · saam vorentoe · masiye phambili

Outline

Projection Pursuit

- (Usually) linear dimension reduction or feature extraction
- Often unsupervised: data exploration
- Look for “interesting projections” of a given set of data
- What is interesting?
- What’s your objective?
  - Clustering? clusters are interesting
  - Outlier detection? outliers are interesting
  - Regression? Derived covariates with a close predictive relationship to the response are interesting
  - Classification? simple boundaries between classes are interesting
  - No idea? we can help you too
    - High information projections

Projection Pursuit

- Broadly speaking, from a mathematical point of view:

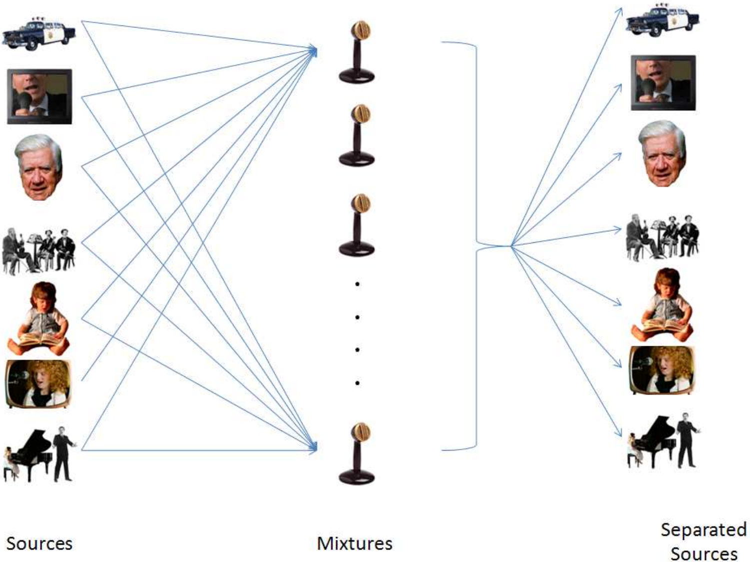
$$\max_{\mathbf{V} \in \mathbb{R}^{p \times p'}} \Phi(\mathbf{V}|\mathbf{X}),$$

  where Φ measures the "interestingness" of the projected data $\mathbf{XV}$.
- Ultimately we are interested in being able to compute some measure of interestingness (whatever that means) of the distribution of the projected data
- The more detail we estimate in this distribution, the more complex our measures can be
  - Second moment: PCA
  - Third moment ...
  - Fourth moment:
    - minimise $\rightarrow$ clustering
    - maximise $\rightarrow$ outliers
  - etc.

Projection Pursuit

- at the extreme: exact estimates of arbitrary functionals of $F_{\mathbf{Xv}}$ or $F_{Y|\mathbf{Xv}}$
- I live at this extreme
- (Very fortunately) frequently the objective can be decomposed over the different *components*, $\mathbf{Xv}_1, ..., \mathbf{Xv}_{p'}$
  - or we force it to decompose since the optimisation problem is considerably easier
- We still are faced with a computational problem:
  - Estimate arbitrary functional from the distribution of univariate (projected) data $\mathbf{Xv}$
  - Compute the gradient of this functional w.r.t. $\mathbf{v}$
  - Repeatedly, over multiple *projection vectors*, $\mathbf{v}^{(0)}, \mathbf{v}^{(1)}, \mathbf{v}^{(2)}, ...$ during optimisation

Independent Component Analysis: The Cocktail Party Problem



Independent Component Analysis: The Model

- Identify *source signals* which are observed only indirectly after "mixing"
- Assume $\mathbf{S} \in \mathbb{R}^{n \times k}$ represents realisations of k INDEPENDENT source signals
- Assume observations are given by $\mathbf{X} = \mathbf{SM}$
- Task is to estimate *unmixing* matrix, $\mathbf{U}$, s.t., $\mathbf{XU} = \mathbf{CPS}$
  - $\mathbf{C}, \mathbf{P}$ are inestimable scaling and permutation matrices
  - We estimate the sources up to arbitrary re-ordering and scalar multiplication
- How could we possibly start to address this?
  - We assume nothing about the sources (except independence)
- We could use projection pursuit to "maximise independence"
  - what do we even mean by this?
  - measuring independence is not usually straight-forward (and is computationally demanding)

Independent Component Analysis: The Method

- Some useful observations massively simplify the problem formulation
  - Orthogonality (zero covariance) is necessary for independence
  - scale doesn't affect independence: $X \perp Y \Rightarrow aX \perp Y \ \forall a$
- We can measure the independence in the components of $\mathbf{XV}$ via their mutual information

$$
\begin{aligned}
MI(\mathbf{XV}) = KL\left(f_{\mathbf{XV}} \middle\| \prod_i f_{\mathbf{Xv}_i} \right) \\
= & E_{Z \sim F_{\mathbf{XV}}} [\log(f_{\mathbf{XV}}(Z))] - E_{Z \sim F_{\mathbf{XV}}} \left[\log \left(\prod_i f_{\mathbf{Xv}_i}(Z_i) \right) \right] \\
= & E_{Z \sim F_{\mathbf{X}}} [\log(f_{\mathbf{X}}(Z))] + \log(|\det(\mathbf{V})|) \\
& - \sum_i E_{Z \sim F_{\mathbf{Xv}_i}} [\log (f_{\mathbf{Xv}_i}(Z))]
\end{aligned}
$$

for non-singular $\mathbf{V}$.

Independent Component Analysis

- But $E_{Z \sim F_{\mathbf{X}}}[\log(f_{\mathbf{X}}(Z))]$ is constant, and we don't care about scale (we can't discriminate based on scale), so can force $\det(\mathbf{V})$ to be constant
- We therefore want to minimise

$$-\sum_i E_{Z \sim F_{\mathbf{X}\mathbf{v}_i}}[\log(f_{\mathbf{X}\mathbf{v}_i}(Z)))]$$

- We don't know $f_{\mathbf{X}\mathbf{v}_i}$:
  - we assume we have a sample from $F_{\mathbf{X}}$ for estimation
- minimise the sample estimate

$$-\sum_i \frac{1}{n} \sum_{j=1}^{n} \log(\hat{f}_{\mathbf{X}\mathbf{v}_i}(\mathbf{x}_j^\top \mathbf{v}_i))$$

- Notice that this is also the maximum (pseudo) likelihood solution, under the assumption of independence

A Quick Aside on Information

- A random variable carrying a lot of information is one which "has a tendency to arise in high density regions"
  - Why is this "information" / "informative"
  - Has a tendency to "show you" where independent copies of itself are likely to "land"
- ALL random variables have "tendency to arise in high density regions"
- Some to a greater extent: For non-decreasing $m : \mathbb{R} \to \mathbb{R}$,

$$E_X[m(f_X(X))] \text{ large } \Rightarrow \text{ Information}(X) \text{ large.}$$

- ICA $\equiv$ maximise the Shannon information of the projections

Independent Component Analysis: The Computation

- Computational problem: Estimating $f_{\mathbf{X}\mathbf{v}}$, and evaluating it at all $\mathbf{x}_j^\top \mathbf{v}$ is expensive if approached naïvely
  - Approximations from "negentropy" (measure of departure from Gaussianity)
  - ... or... or
  - or suck it up and get better at computation
- To be completely general, we must be non-parametric
  - I like kernels, they're more intuitive (to me) and amenable to differentiation
- The kernel estimate of a density, f_X, using a sample from its distribution, $\{x_1, ..., x_n\}$, is given by

$$\hat{f}(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)$$

Fast Kernel Smoothing

- We'll be slightly more general, and look at evaluating

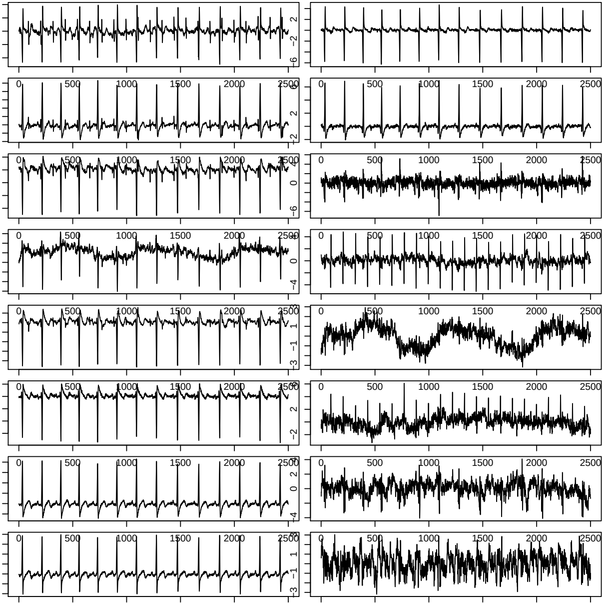
$$\hat{f}(x) = \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)\omega_i,$$

  for arbitrary $\{\omega_1, ..., \omega_n\}$
- To evaluate directly, costs $\mathcal{O}(nm)$ to evaluate at m points
- But, if the kernels look like this:

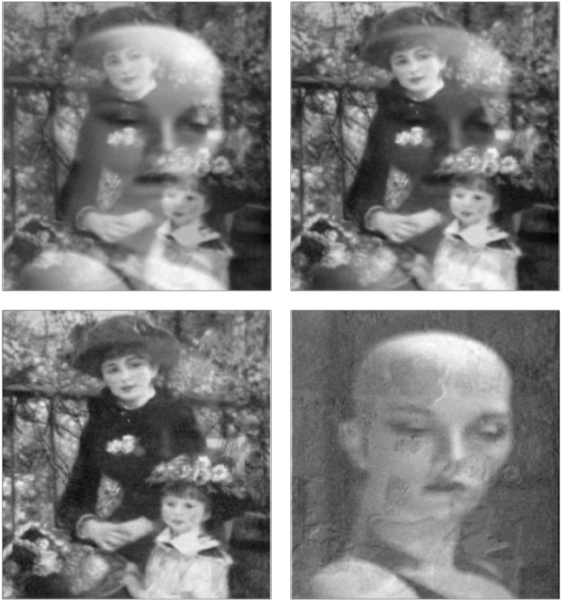
$$K(x) = \sum_{i=0}^{\alpha} \beta_i |x|^i \exp(-|x|)$$

  we can do much better $(\mathcal{O}(n \log(n))$
- Fast computation boils down to:
  1. trivial factorisation of $\exp(a + b)$
  2. binomial expansion of $(a + b)^i$

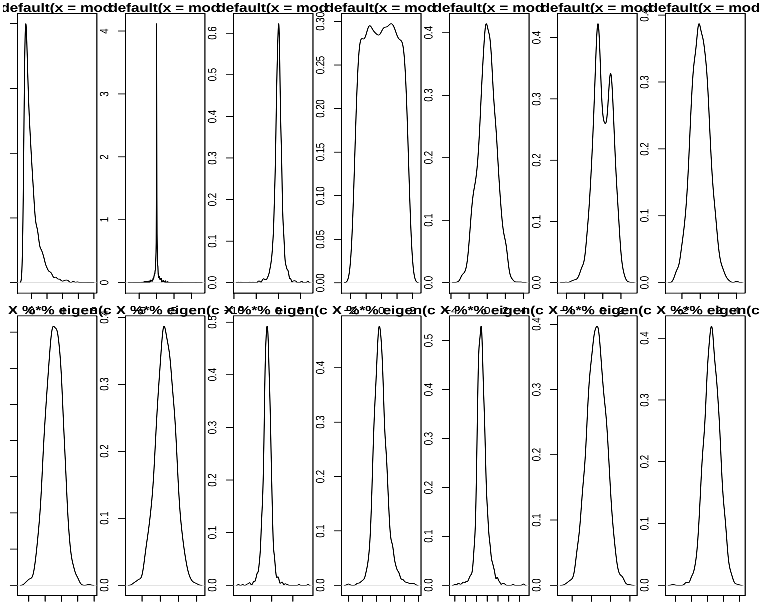
A Few ICA Successes: Foetal ECG



A Few ICA Successes: Reflection removal



ICA: What's it actually doing? (vs PCA)



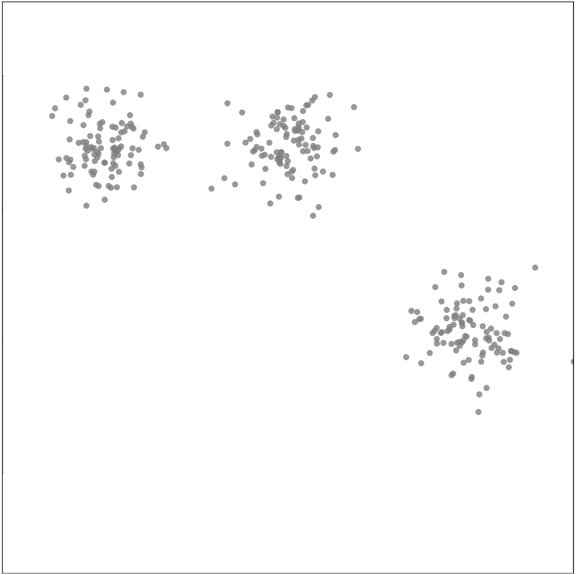
Other Examples: Projection Pursuit for Clustering

- Maximise over $\mathbf{v}$,

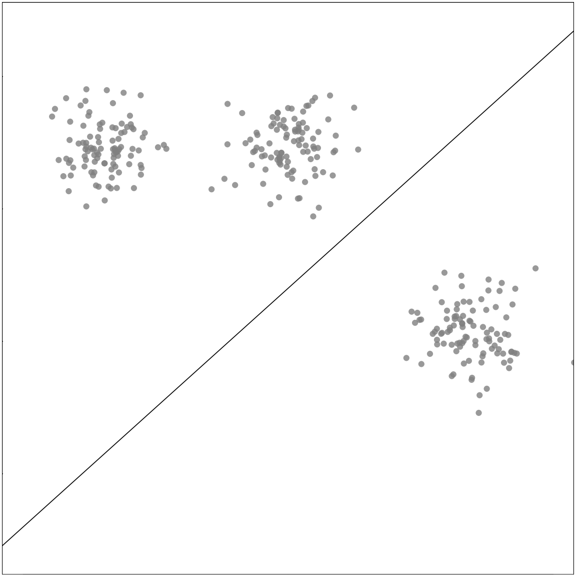
$$\Phi(\mathbf{v}|\mathbf{X}) = \max_{b} Q(\mathbf{v}, b|\mathbf{X}),$$

  where Q measures the quality of clustering $\mathbf{X}$ using hyperplane $H(\mathbf{v}, b) = \{\mathbf{x} \in \mathbb{R}^p | \mathbf{x}'\mathbf{v} = b\}$.
- Examples:
  - Minimise surface integral of density on $H(\mathbf{v}, b)$
  - Minimise normalised cut across $H(\mathbf{v}, b)$
  - Maximise variance ratio of clusters (similar to LDA objective)

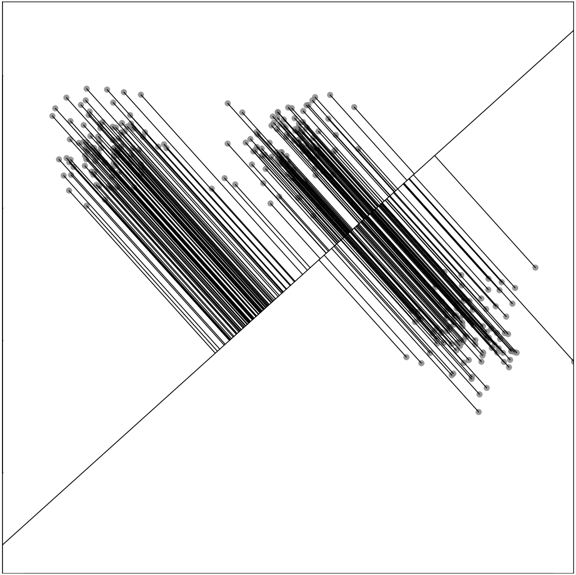
Other Examples: Projection Pursuit for Clustering (in action)



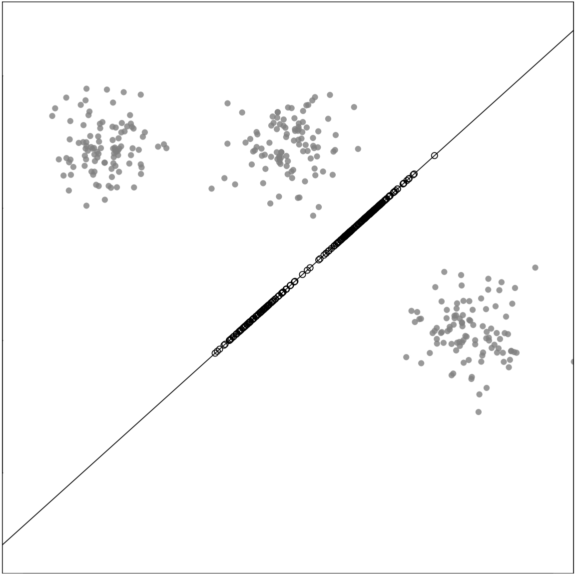
Other Examples: Projection Pursuit for Clustering (in action)



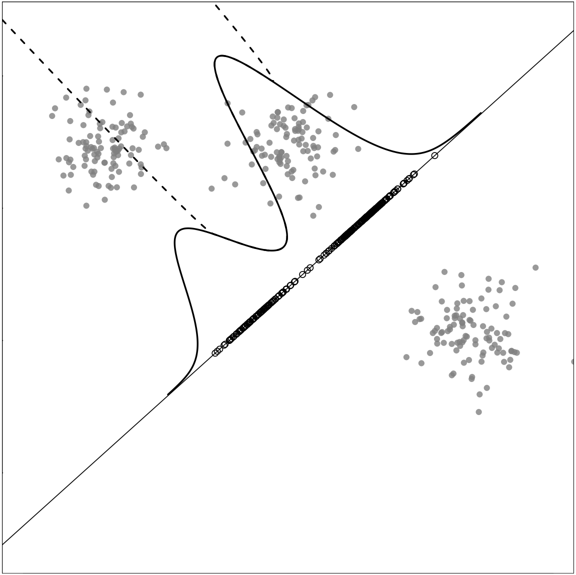
Other Examples: Projection Pursuit for Clustering (in action)



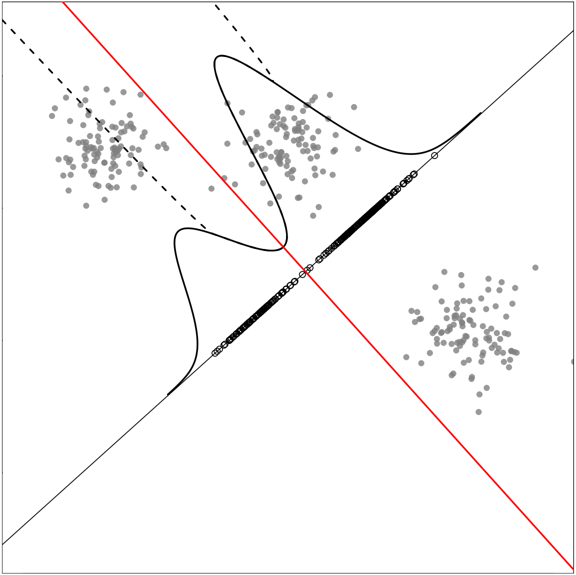
Other Examples: Projection Pursuit for Clustering (in action)



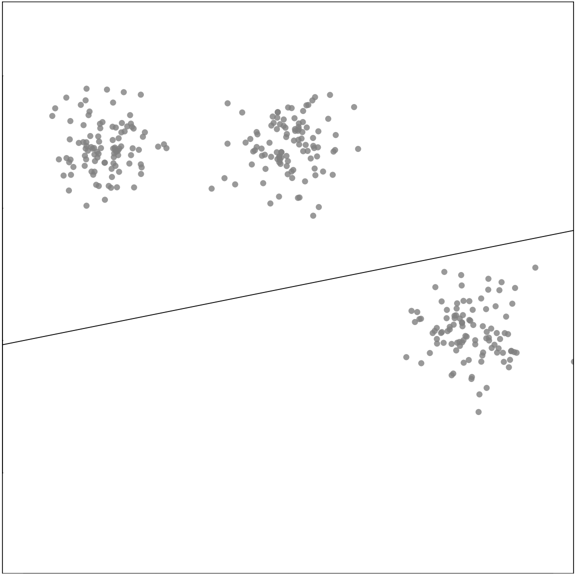
Other Examples: Projection Pursuit for Clustering (in action)



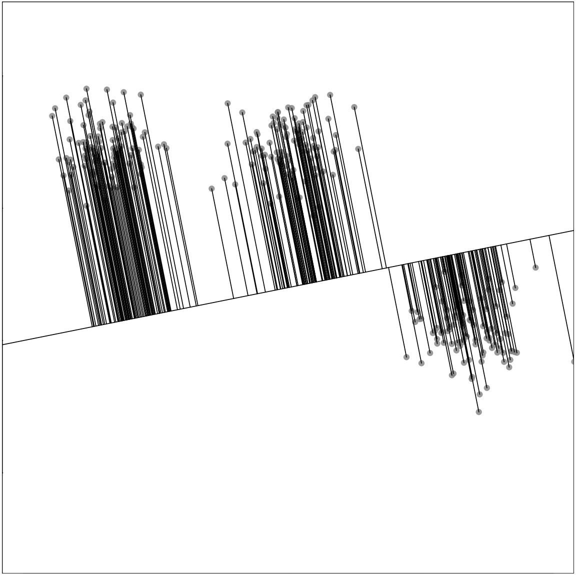
Other Examples: Projection Pursuit for Clustering (in action)



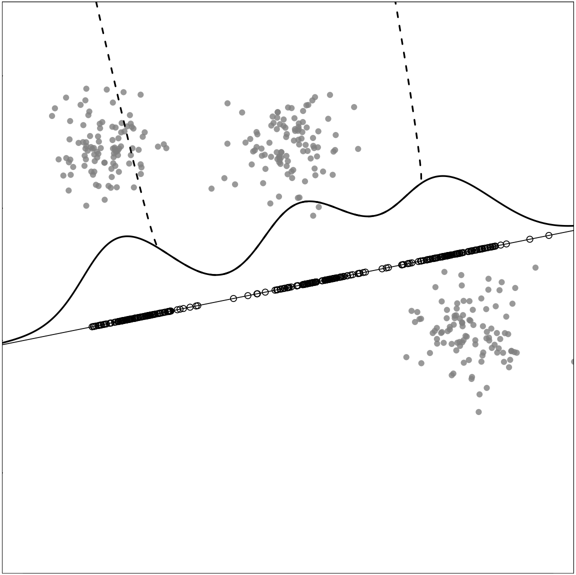
Other Examples: Projection Pursuit for Clustering (in action)



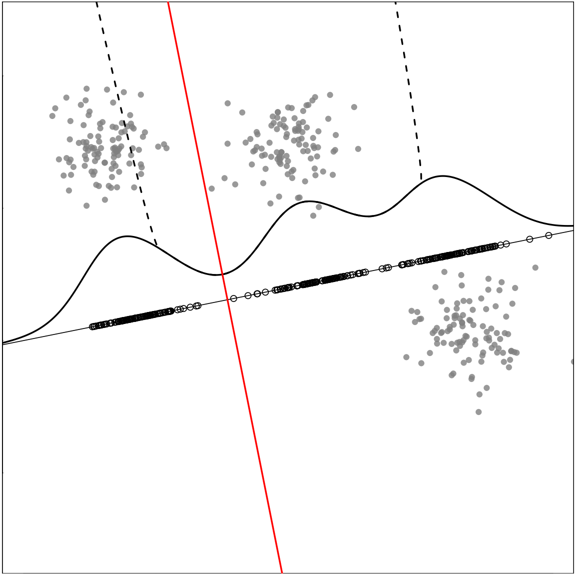
Other Examples: Projection Pursuit for Clustering (in action)



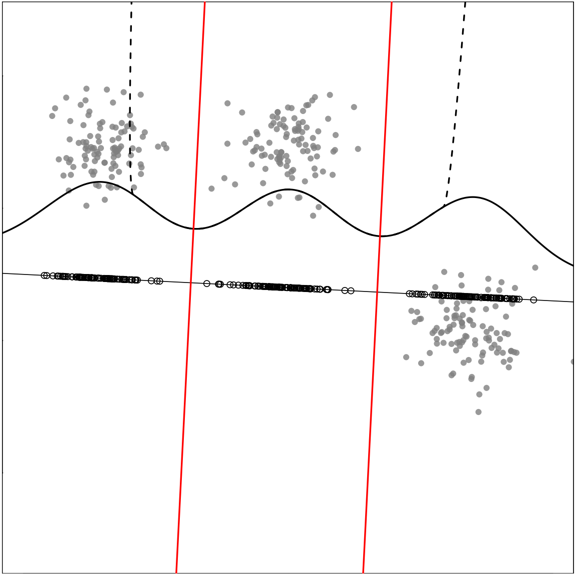
Other Examples: Projection Pursuit for Clustering (in action)



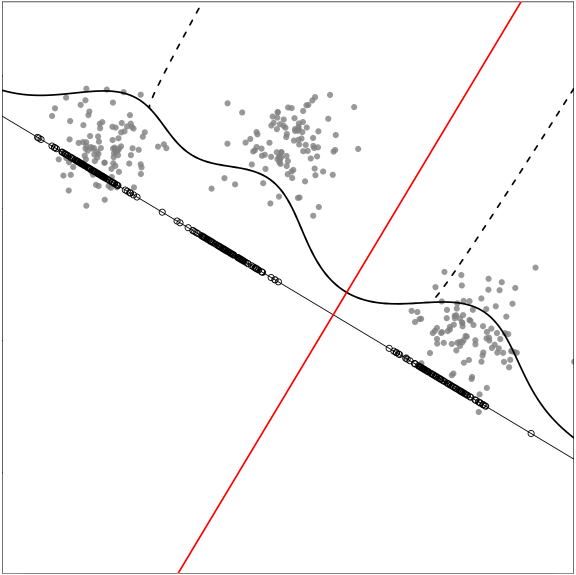
Other Examples: Projection Pursuit for Clustering (in action)



Other Examples: Projection Pursuit for Clustering (in action)



Other Examples: Projection Pursuit for Clustering (in action)



Other Examples: Projection Pursuit Regression

- Additive model with "ridge" functions
- GAM with derived features
- single layer NN with non-parametric activation function
- Estimate $f(\mathbf{x}) = E[Y|\mathbf{X} = \mathbf{x}]$ by minimising

$$\sum_{i=1}^{n} \mathcal{L}\left(y_i, \mu + \sum_{j=1}^{k} g_j(\mathbf{x}_i^\top \mathbf{v}_j)\right)$$

Other Examples: Projection Pursuit Regression

- Each g_j is fit non-parametrically (e.g., with kernels)
- For notational brevity: $S_j(x|\mathbf{b}) = \sum_{i=1}^n K\left(\frac{\mathbf{x}_i^\top \mathbf{v}_j - x}{h}\right) b_i$
- Local constant (Nadaraya-Watson):

$$\hat{g}_j(x) = \frac{\sum_{i=1}^n K\left(\frac{\mathbf{x}_i^\top \mathbf{v}_j - x}{h}\right) y_i}{\sum_{i=1}^n K\left(\frac{\mathbf{x}_i^\top \mathbf{v}_j - x}{h}\right)} = \frac{S_j(x|\mathbf{y})}{S_j(x|\mathbf{1})}$$

- Local linear:

$$\begin{aligned}\hat{g}_j(x) =& \frac{S_j(x|(\mathbf{X}\mathbf{v}_j)^2)S_j(x|\mathbf{y}) - S_j(x|\mathbf{X}\mathbf{v}_j)S_j(x|\mathbf{X}\mathbf{v}_j * \mathbf{y})}{S_j(x|(\mathbf{X}\mathbf{v}_j)^2)S_j(x|\mathbf{1}) - S_j(x|\mathbf{X}\mathbf{v}_j)^2} \\ &+ \frac{S_j(x|\mathbf{1})S_j(x|\mathbf{X}\mathbf{v}_j * \mathbf{y}) - S_j(x|\mathbf{X}\mathbf{v}_j)S_j(x|\mathbf{y})}{S_j(x|(\mathbf{X}\mathbf{v}_j)^2)S_j(x|\mathbf{1}) - S_j(x|\mathbf{X}\mathbf{v}_j)^2} x\end{aligned}$$

Other Examples: Optimal Projections for NB

- Naïve Bayes has enjoyed a lot of success, despite its very simplistic model: Classify based on standard

$$P(Y = k|\mathbf{X} = \mathbf{x}) = \frac{f_{\mathbf{X}|Y=k}(\mathbf{x})P(Y = k)}{\sum_{j=1}^{K} f_{\mathbf{X}|Y=j}(\mathbf{x})P(Y = j)}$$

  but treat $f_{\mathbf{X}|Y=k}$ as though
  - Treat covariates as being completely independent, conditional on the class label
  - In the Gaussian case, like diagonal discriminant analysis
  - By treating these as independent, we can fit more flexible models (e.g., non-parametric ones) easily
- Can we find a projection $\mathbf{V}$, so that NB applied to $\mathbf{XV}$ fits well?
  - Like changing the basis for factorisation of the densities

Other Examples: Optimal Projections for NB

- An earlier approach applied ICA to each class separately (CCICA) to better approximate the independence assumptions: Hopefully less bias in the estimate of each $f_{\mathbf{X}|Y=k}$
  - May not aid in better discrimination
- Maximise the classification likelihood (multinomial likelihood, conditional on fixed $\mathbf{X}$), with a single projection matrix

$$\prod_{i=1}^{n} \frac{\hat{f}_{\mathbf{XV}|Y=y_i}(\mathbf{x}_i^\top \mathbf{V})\hat{P}(Y = y_i)}{\sum_{j=1}^{K} \hat{f}_{\mathbf{XV}|Y=j}(\mathbf{x}_i^\top \mathbf{V})\hat{P}(Y = j)}$$

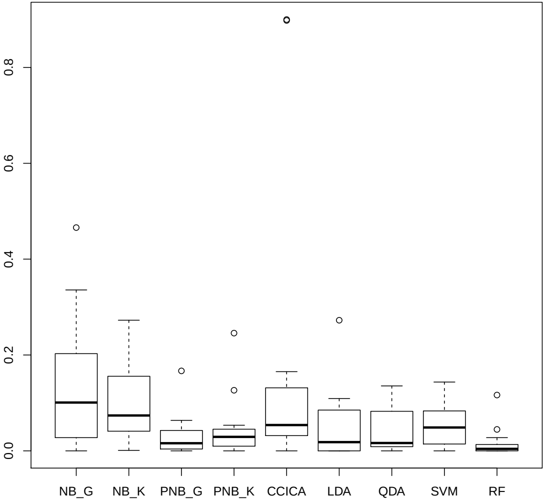
Other Examples: Optimal Projections for NB

- (Obviously) we log this, and then do some simple algebra, and find the objective is given by

$$\sum_{k=1}^{K}\left(\log\left(\frac{n_k}{n}\right)+\sum_{j:y_j=k}\log\left(\hat{f}_{\mathbf{XV}|Y=k}(\mathbf{x}_j^\top\mathbf{V})\right)\right) \\ -\sum_{j=1}^{n}\log\left(\sum_{k=1}^{K}\frac{n_k}{n}\hat{f}_{\mathbf{XV}|Y=k}(\mathbf{x}_i^\top\mathbf{V})\right)$$

- The first term (after factorising the densities) is a weighted sum of the ICA objectives for the classes, and the second term penalises for non-discrimination
- This can similarly be written using combinations of kernel weighted sums

Some Plots (vs Others)



Playing with ICA in `R`

- We will implement our own ICA method in `R` using kernel estimates for the densities
- The package `FKSUM` provides implementations of the log-linear time methods
- We will use a similar approach to that used in PCA based on projection pursuit, where projection of the argument onto the unit ball forms part of the evaluation: If $\Phi(\mathbf{v}|\mathbf{X}) = \phi(\mathbf{X}\vec{\mathbf{v}})$ is the objective function, then

$$\nabla_{\mathbf{v}}\Phi(\mathbf{v}|\mathbf{X}) = \frac{1}{||\mathbf{v}||}(I - \vec{\mathbf{v}}\vec{\mathbf{v}}')\mathbf{X}'\nabla_{\mathbf{p}}\phi(\mathbf{p})\Big|_{\mathbf{p}=\mathbf{X}\vec{\mathbf{v}}}$$

Playing with ICA in R

- We have

$$\phi(\mathbf{p}) = -\frac{1}{n}\sum_{i=1}^{n}\log(\hat{f}_{\mathbf{p}}(p_i)) = -\frac{1}{n}\sum_{i=1}^{n}\log\left(\frac{1}{nh}\sum_{j=1}^{n}K\left(\frac{p_i - p_j}{h}\right)\right)$$

$$\Rightarrow \frac{\partial}{\partial p_k}\phi(\mathbf{p}) = \frac{1}{n^2h^2}\sum_{i=1}^{n}K'\left(\frac{p_i - p_k}{h}\right)\left(\frac{1}{\hat{f}_{\mathbf{p}}(p_i)} + \frac{1}{\hat{f}_{\mathbf{p}}(p_k)}\right)$$