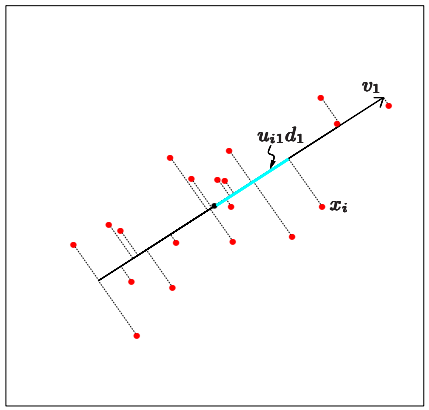
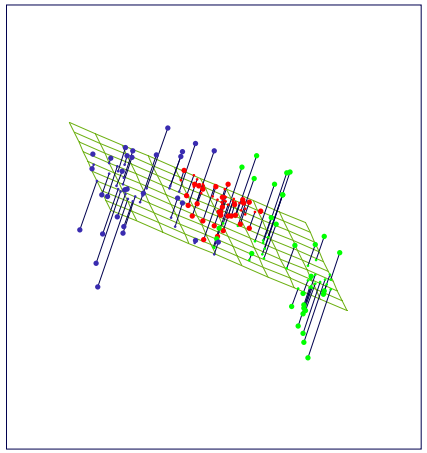
Further ML Methods

- Linear Projections
  - **Principal Component Analysis (PCA)**: 'the most spread variance' directions
    - **Sparse PCA**. (sklearn)
  - **Partial Least Squares**: not mentioned here. (sklearn)
  - **Archetypal analysis**: extremes, instead of 'centers' from clustering; data=lin. comb. of archetypes (archetypes)
  - **NMF Nonnegative Matrix Factorization**: 'linear r-dimensional autoencoder' (sklearn)
  - **Factor analysis**: A view on 'independent factors' observed via a linear combination mixture with a gaussian noise (sklearn)
  - **Independent Component Analysis**: splits the signal according to non-gaussian features (max. divergence from gaussian) (sklearn)
  - **Procrustes transformation** - curve fitting.
- **Principal curves and surfaces** (predefined $f_j(\lambda)$, curve paramater λ) (prinPy),
- **Kernel PCA**. (sklearn)
- **Spectral Clustering**. (sklearn)

PCA Principal Components, Curves and Surfaces

- The **principal components** of a set of data in $\mathbb{R}^p$ provide a sequence of best linear approximations to that data, of all ranks $q \leq p$.
- let μ be a location vector in $\mathbb{R}^p$, V_q is a $p \times q$ matrix with q orthogonal unit vectors as columns, λ is a q vector of parameters.
- $f(\lambda) = \mu + V_q\lambda$ represents an affine hyperplane of rank q.
- We minimize the **reconstruction error** (by least squares)
  - $min_{\mu,\{\lambda_i\}} \sum_{i=1}^{N} \|x_i - \mu - V_q\lambda_i\|^2$.
- We can partially optimize
  - $\hat{\mu} = \overline{x}$
  - $\hat{\lambda}_i = V_q^T(x_i - \overline{x})$.
- This leaves us to find the **orthogonal matrix** V_q $min_{V_q} \sum_{i=1}^{N} \left\|(x_i - \overline{x}) - V_q V_q^T(x_i - \overline{x})\right\|^2$.
- We center the data $\overline{x} = 0$ to simplify the formulas.

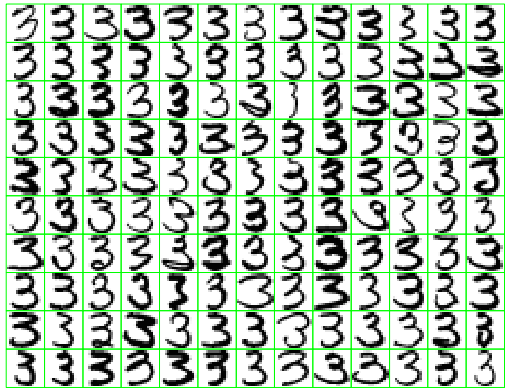


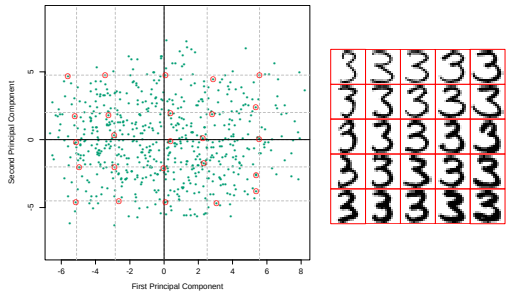


Handwritten 3 Example

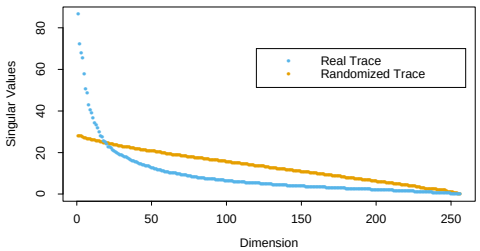
- 130 handwritten digits 3, each 16×16 grayscale image.
  - $x \in \mathbb{R}^{256}$
- First two principal component plot
  - For the first two principal components quantiles 5,25,50,75,95 percent.
  - First component - x axis: mainly the length of 3
  - Second component - the thickness.
- The projection on the first two components is:

$$\begin{aligned} \hat{f}(\lambda) &= \bar{x} + \lambda_1 v_1 + \lambda_2 v_2 \\ &= \bar{x} + \lambda_1 \cdot v_1 + \lambda_2 \cdot v_2. \end{aligned}$$





- First 12 components account for 63% data variations.
- Explained variance by PCA (blue) and randomized directions (orange).



Sparse Principal Components

- We often interpret PCA by examining **loadings**: direction vectors v_j.
- This interpretation is easier if the loadings are sparse.

Definition (Sparse PCA)

Sparse principal component technique solves for a single component:

$$min_{\theta,v} \sum_{i=1}^{N} \|x_i - \theta v^T x_i\|_2^2 + \lambda \|v\|_2^2 + \lambda_1 \|v\|_1$$

$$\text{subject to } \|\theta\|_2 = 1.$$

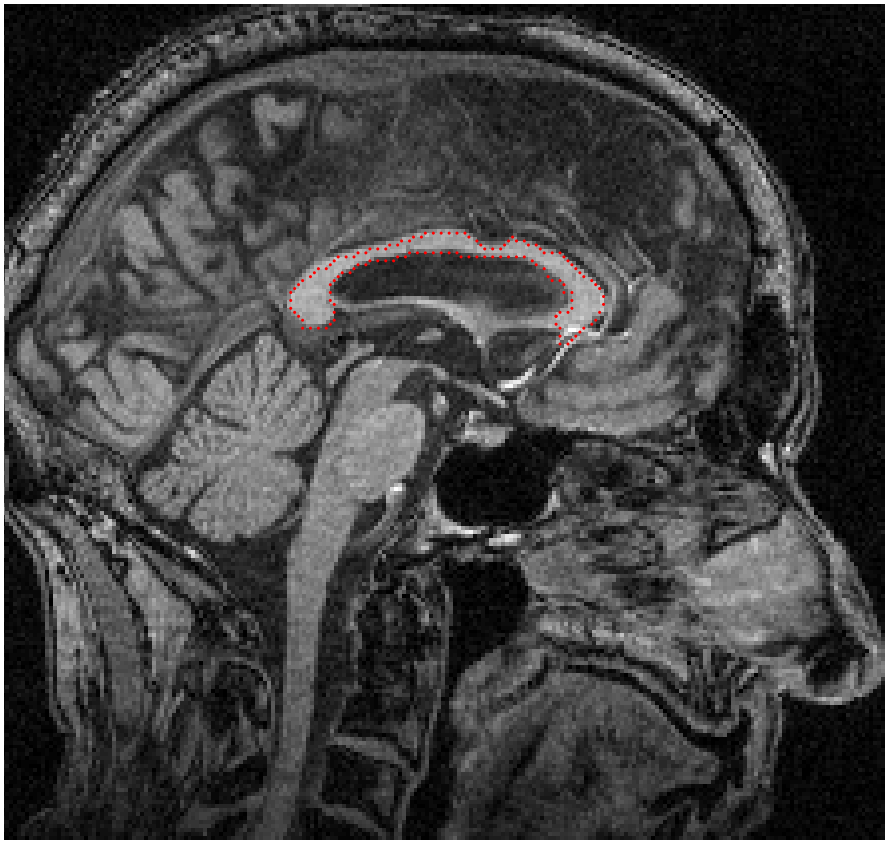
- If both $\lambda = \lambda_1 = 0$ and $N > p$, than $v = \theta$ is the largest principal component direction.
- When $p \gg N$ the solution may not be unique unless $\lambda > 0$. For $\lambda > 0$ and $\lambda_1 = 0$ is the solution proportional to the largest principal component direction.

Sparse principal component for multiple components minimizes Θ and V $p \times K$ matrices

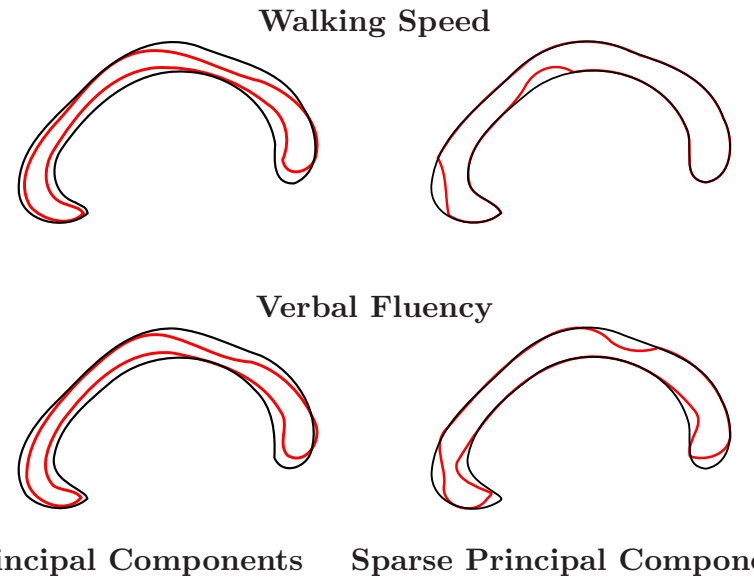
$$min_{\Theta,V} \sum_{i=1}^{N} \|x_i - \Theta V^T x_i\|_2^2 + \lambda \sum_{k=1}^{K} \|v_k\|_2^2 + \sum_{k=1}^{K} \lambda_{1k} \|v_k\|_1$$

$$\text{subject to } \Theta^T \Theta = I_K.$$

Corpus Callosum (CC) Sparse PCA Example



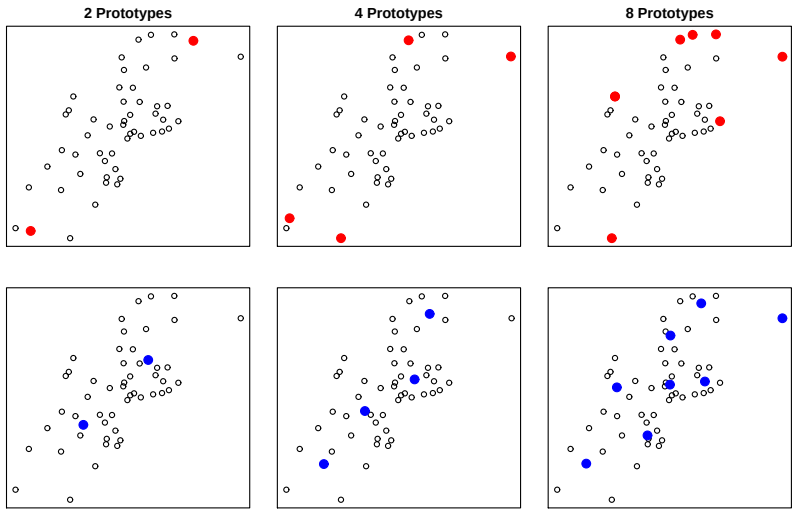
- The Corpus Callosum scan.
- The area represented by a number of points aligned by Procrustes analysis,
- a set of 2d points for now.



Archetypal Analysis

- **Archetypal Analysis** approximates data points by a linear combination of prototypes
  - that are themselves linear combinations of data points.
  - Each data point is approximated by a convex combination of prototypes.
  - This forces the prototypes to lie on the convex hull of the data cloud.
  - In this sense, they are 'archetypal'.

- K-means clustering
  - approximates any point by one prototype
  - each prototype is a linear combination of samples (the mean of a cluster).



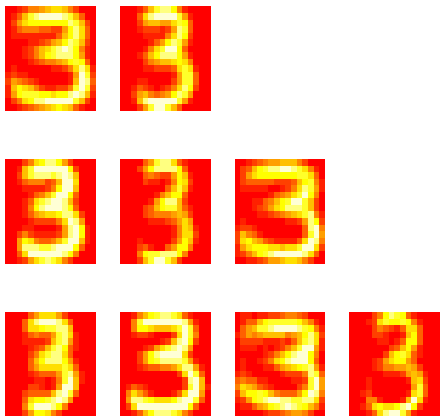
Archetypal Analysis

- "linear autoencoder" of the dimension r

Definition (Archetypal Analysis)

- A non-negative $N \times p$ data matrix X is modeled $X \sim WH$,
  - $H = BX$ is $r \times p$ matrix of r **archetypes** (rows of H),
  - B is $r \times N$ matrix where $b_{ki} \geq 0$ and $(\forall k)$ $(\sum_{i=1}^{N} b_{ki} = 1)$.
  - W is $N \times r$ matrix where $w_{ik} \geq 0$ and $(\forall i)$ $(\sum_{k=1}^{r} w_{ik} = 1)$.
  - We minimize over W and B: $J(W, B) = \|X - WH\|^2 = \|X - WBX\|^2$.

- Its minimized in an alternating fashion, with each separate minimization involving a convex optimization.
- Converges to a local minimum.
- Figure: 2,3, and 4 prototypes for the Handwritten 3 example.
- Extreme 3's both in size and shape.

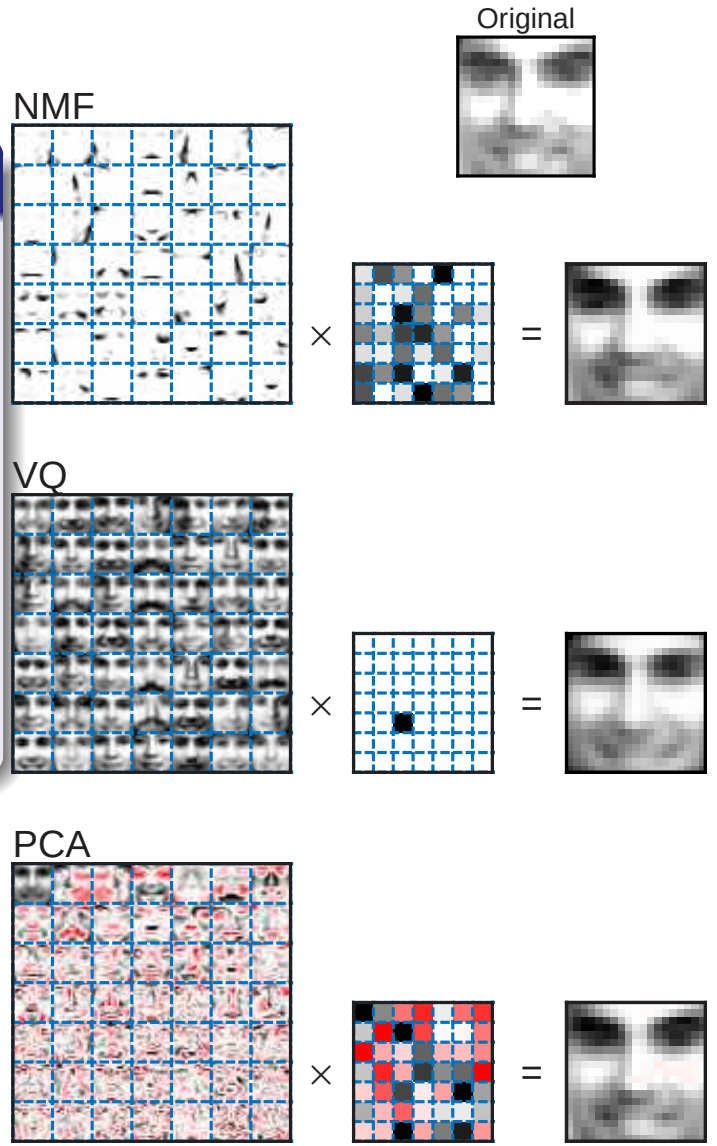


Non-negative Matrix Factorization

Definition (Non-negative Matrix Factorization)

- A centered $N \times p$ data matrix X is modeled $X \sim WH$,
- W is $N \times r$ matrix, H is $r \times p$, $r \leq \max(N, p)$.
- We assume $x_{ij}, w_{ik}, h_{kj} \geq 0$.
- We maximize over W and H: $L(W, H) = \sum_{i=1}^{N} \sum_{j=1}^{p} [x_{ij} \log(WH)_{ij} - (WH)_{ij}]$.

- NMF assumes x_{ij} has a Poisson distribution with mean $(WH)_{ij}$
- we maximize the loglikelihood.



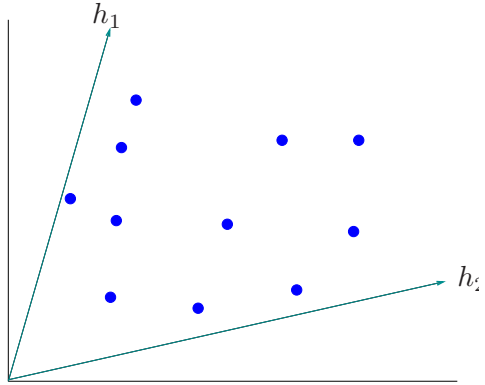
NMF

NMF

1: **procedure** NMF:(X centered data)
2: **repeat**
3: $$w_{ik} \leftarrow w_{ik} \frac{\sum_{j=1}^{p} h_{kj} x_{ij} / (WH)_{ij}}{\sum_{j=1}^{p} h_{kj}},$$
4: $$h_{kj} \leftarrow h_{kj} \frac{\sum_{i=1}^{N} w_{ik} x_{ij} / (WH)_{ij}}{\sum_{i=1}^{N} b_{ik}}$$
5: **until** convergence
6: return W, H
7: **end procedure**

- The NMF solution is not unique.

Any h_1, h_2 basis vectors in the open space between the coordinate axes and data work (given an exact reconstruction of the data).



sklearn.decomposition.NMF has the objective function:

$$0.5\|X - WH\|_{loss}^2 + \alpha_W \cdot l1_{ratio} p \|vec(W)\|_1 + 0.5\alpha_W \cdot (1 - l1_{ratio}) p \|W\|_{Fro}^2 + \alpha_H \cdot l1_{ratio} N \|vec(H)\|_1 + 0.5\alpha_H \cdot (1 - l1_{ratio}) N \|H\|_{Fro}^2$$

- $\|vec(W)\|_1 = \sum_{i,j} abs(W_{i,j})$ elementwise L1 norm
- $\|W\|_{Fro}^2 = \sum_{i,j} W_{i,j}^2$ Frobenius norm
- *loss* is Frobenius norm or another beta-divergence loss, $l1_{ratio} = 0$.

Independent Component Analysis

- Multivariate data as multiple indirect measurements from an underlying source.
- Examples: EEG brain scans, 'body fat', trading prices.
- **Factor analysis**
  - typically wed to Gaussian distributions
  - which has hindered their usefulness
  - and has no unique solution
    - any linear transformation is a solution.

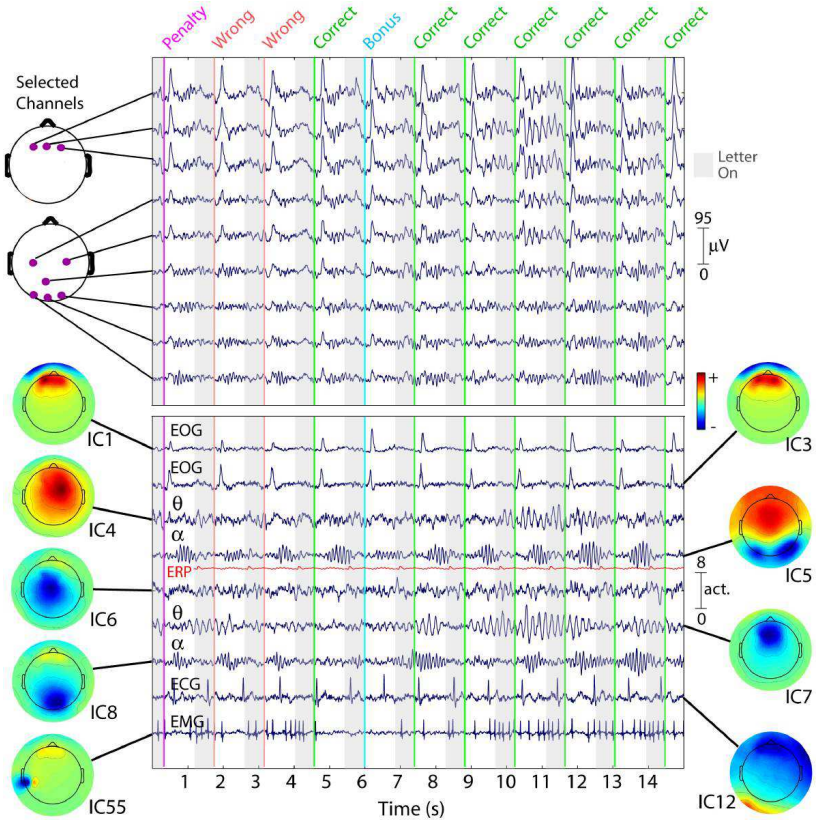


FIGURE 14.41. *Fifteen seconds of EEG data (of 1917 seconds) at nine (of 100) scalp channels (top panel), as well as nine ICA components (lower panel). While nearby electrodes record nearly identical mixtures of brain and non-brain activity, ICA components are temporally distinct. The colored scalps represent the ICA unmixing coefficients $\hat{a}_j$ as a heatmap, showing brain or scalp location of the source.*

Latent Variables and Factor Analysis

- Take the singular value decomposition $X = UDV^T$
  - we assume that the columns of X have zero mean
  - where D is a diagonal matrix
  - U is orthogonal.
- X has a **latent variable decomposition** $X = SA^T$
  - where $S = \sqrt{N}U$, $A^T = \frac{1}{\sqrt{N}}DV^T$
  - each of the columns of X is a linear combination of the columns of S
  - columns of S have zero mean, are uncorrelated and have unit variance, $Cov(S) = I$.
  - we can interpret the SVD, or the corresponding PCA as an estimate of a latent variable model $X = AS$

$$
\begin{aligned}
X_1 &= a_{11}S_1 + a_{12}S_2 + \ldots + a_{1p}S_p \\
X_2 &= a_{21}S_1 + a_{22}S_2 + \ldots + a_{2p}S_p \\
\ldots & \\
X_p &= a_{p1}S_1 + a_{p2}S_2 + \ldots + a_{pp}S_p
\end{aligned}
$$

- Notice that for any orthogonal $p \times p$ matrix R is $X = AS = AR^T RS = A^* S^*$.

Factor Analysis

- In the SVD decomposition any rank $q < p$ truncated decomposition approximates X in an optimal way.
- **Factor analysis model** (popular in psychometrics)
  - with $q < p$, a factor analysis model has the form $X = AS + \epsilon$

$$\begin{aligned} X_1 &= a_{11}S_1 + a_{12}S_2 + \ldots + a_{1q}S_q + \epsilon_1 \\ X_2 &= a_{21}S_1 + a_{22}S_2 + \ldots + a_{2q}S_q + \epsilon_2 \\ \ldots & \\ X_p &= a_{p1}S_1 + a_{p2}S_2 + \ldots + a_{pq}S_q + \epsilon_p \end{aligned}$$

  - S is a vector of $q < p$ underlying latent variables or factors
  - A is a $p \times q$ matrix of factor **loadings**
    - used to name and interpret the factors
  - ϵ_j are uncorrelated zero–mean disturbances.
  - Typically, S_ℓ and ϵ_j are modeled as Gaussian random variables, and the model is fit by maximum likelihood.
  - The parameters all reside in the covariance matrix

$$\Sigma = AA^T + D_\epsilon$$

  - where $D_\epsilon = diag[Var(\epsilon_1), Var(\epsilon_2), \ldots, Var(\epsilon_p)]$
  - S independent factors like intelligence, drive in a battery of educational tests.

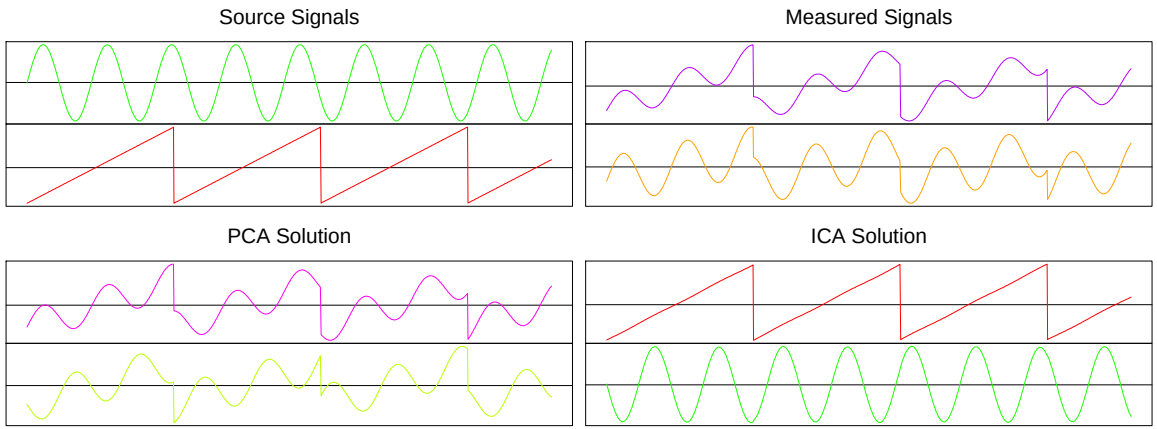
Independent Component Analysis

$$
\begin{aligned}
X_1 &= a_{11}S_1 + a_{12}S_2 + \ldots + a_{1p}S_p \\
X_2 &= a_{21}S_1 + a_{22}S_2 + \ldots + a_{2p}S_p \\
&\ldots \\
X_p &= a_{p1}S_1 + a_{p2}S_2 + \ldots + a_{pp}S_p
\end{aligned}
$$

- S are assumed **statistically independent** rather than uncorrelated
  - correlation: second order interaction
  - independence: all orders of interactions.
  - Multivariate Gaussian is determined by its second moments alone (up to rotation).
  - Otherwise, the extra moments allow to identify the elements of A uniquely.
- We assume X has been **whitened** to have $Cov(X) = I$; Simplest: multiply by $W = \Sigma^{-\frac{1}{2}}$, typically achieved via the SVD to $D^{-\frac{1}{2}}V^T$.
  - $Var(S) = I$, therefore is A orthogonal.
  - ICA searches an orthogonal S such that $S = A^T X$ are independent (not-Gaussian) components.

Example

- **Cocktail party problem** Different microphones X_j pick up mixtures of different independent sources S_ℓ (music, speech from different speakers).
- ICA is able to perform blind source separation
  - by exploiting the independence and non–Gaussianity of the original sources.



Entropy

$$
\begin{aligned}
\text{entropy } H(Y) &= -\int g(y) \log g(y) dy \\
\text{mutual information } I(Y) &= \sum_{j=1}^{p} H(Y_j) - H(Y) \\
I(Y) &= \sum_{j=1}^{p} H(Y_j) - H(X) - \log |\det(A)| \\
&= \sum_{j=1}^{p} H(Y_j) - H(X)
\end{aligned}
$$

- since $Cov(X) = I$, $Y = A^T X$ and A is orthogonal.
- We search A to minimize $I(Y) = I(A^T X)$
  - looks for the orthogonal transformation that leads to the most independence between its components
  - minimizes the sum of the entropies of the separate components of Y
  - this amounts to maximizing their departures from Gaussianity.

Negentropy, FastICA

- For each Y_j, let Z_j be a Gaussian random variable with the same variance as Y_j.
- The **negentropy** $J(Y_j)$ is defined

$$J(Y_j) = H(Z_j) - H(Y_j)$$

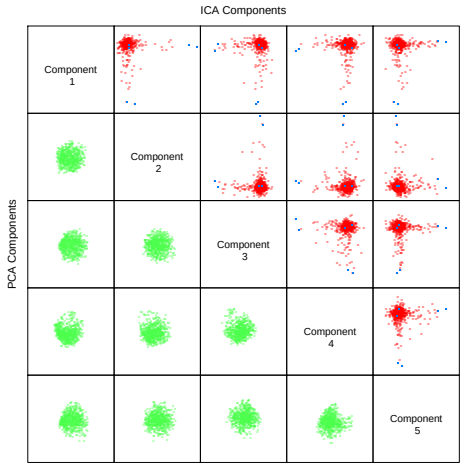
- It is non–negative, and measures the departure of Y_j from Gaussianity.
- Can be approximated by

$$J(Y_j) \sim [\mathbb{E}G(Y_j) - \mathbb{E}G(Z_j)]^2$$

  - $G(u) = \frac{1}{a} \log \cosh(au)$ for $1 \leq a \leq 2$.

FastICA

- ICA starts from essentially a factor analysis solution
- and looks for rotations that lead to independent components.



- Above diagonal: first five ICA components
- Below diagonal: first five PCA components
- all standardized to unit variance.

FastICA

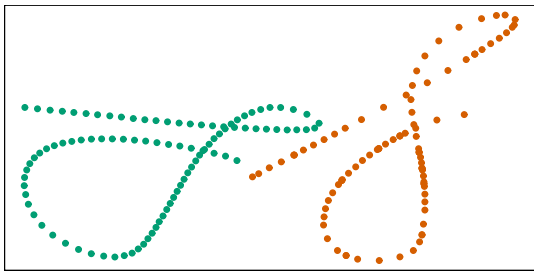
- X are centered and whitened data with $Cov(X) = I$
  - typically achieved via the SVD, $X \leftarrow \sqrt{D}U$ from $X = UDV^T$.
- q the number of components
  - Only one is allowed to follow Gaussian distribution.
- a a parameter.

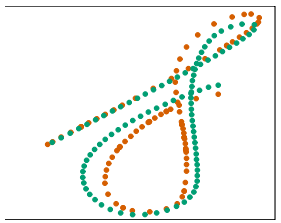
FastICA

procedure FastICA:($\mathbf{X}$, $a \in \langle 1, 2 \rangle$, $q \leq p$)
  $w_1, \ldots, w_q \leftarrow$ randomly initialize N-dimensional weight vectors
  for $\ell = 1, \ldots, q$ **do**
    repeat
      $w_\ell^+ \leftarrow \sum_{i=1}^N x \tanh(a w_\ell^T x) - \left(\sum_{i=1}^N \frac{1}{cosh^2(w_\ell^T x)} \right) w$
      $w_\ell \leftarrow w_\ell^+ - \sum_{j=1}^{\ell-1} w_\ell^T w_j w_j$ # orthogonal to previous
      $w_\ell \leftarrow \frac{w_\ell}{\sqrt{w_\ell^T w_\ell}}$ # normalize
    until convergence
  end for
end procedure

Procrustes Transformations

- Assume each handwritten S is represented as $N = 96$ points.
- Both X_1 and X_2 are $N \times 2$ matrices (green, orange curve).
  - with column means $\overline{x}_1, \overline{x}_2$, centered to $\tilde{X}_1, \tilde{X}_2$.
  - To find landmarks (points) are difficult and subject specific.
  - In this example, dynamic time wrapping of the speed signal along each signature was used.





- **Procrustes** transformation
  - R is an orthonormal $p \times p$ matrix,
    - $\hat{R} \leftarrow UV^T$ from $\tilde{X}_1^T \tilde{X}_2 = UDV^T$.
  - μ a p vector of location coordinates
    - $\mu \leftarrow \overline{x}_2 - \hat{R}\overline{x}_1$.
  - $\|X\|_F^2 = trace(X^T X) = \sum_{i=1}^{N} \sum_{j=1}^{p} |x_{ij}|^2$ is the squared **Frobenius** matrix norm
  - We minimize the **Procrustes distance**

$$min_{\mu,R} \left\| X_2 - (X_1 R + 1\mu^T) \right\|_F^2 .$$

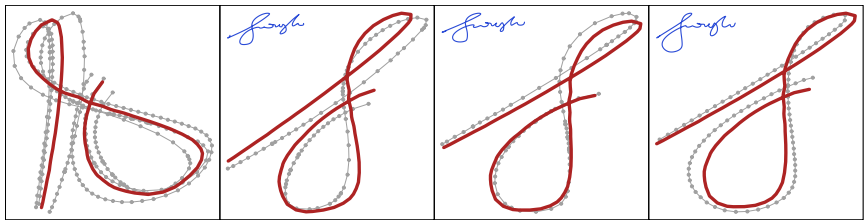
Shape Averaging, Procrustes Average

! From now on, we assume the data are centered.

Definition (Procrustes average)

The **Procrustes average** of a collection of L shapes is M that minimizes

$$min_{\{R_\ell\}_1^L, M} \sum_{\ell=1}^{L} \|X_\ell R_\ell - M\|_F^2 .$$

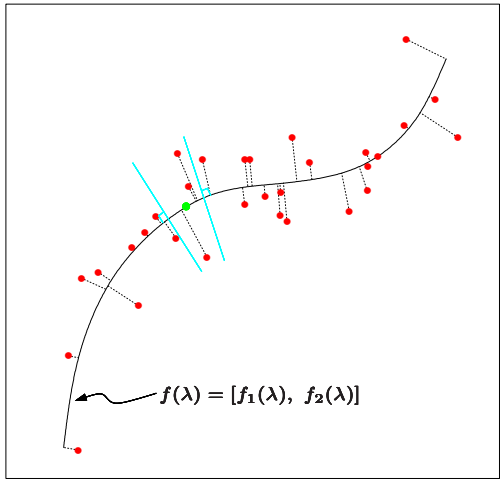


Procrustes Average

1: **procedure** PROCRUSTES AVERAGE:($N \times p$ shapes $\{\mathbf{X}_\ell\}_{\ell=\mathbf{1}}^{\mathbf{L}}$)
2: $M \leftarrow X_1$ # init the average
3: **repeat**
4: $X'_\ell \leftarrow X_\ell \hat{R}_\ell$ # M fixed, solve L Procruster rotations $\hat{R}_\ell$
5: $M \leftarrow \frac{1}{L} \sum_{\ell=1}^{L} X'_\ell$ # average
6: **until** convergence
7: **end procedure**

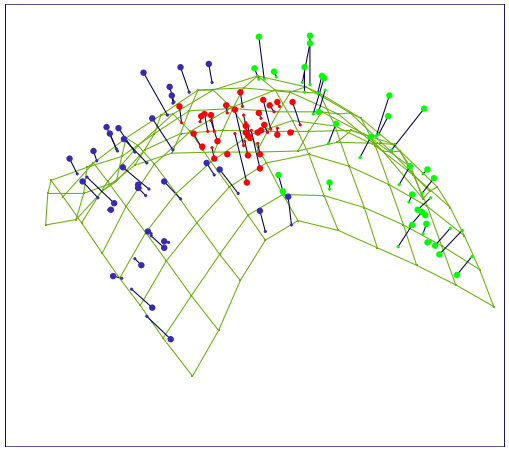
Principal Curves and Surfaces

- To find a principal curve $f(\lambda)$ of a distribution, we consider
- $f(\lambda) = [f_1(\lambda), f_2(\lambda), \ldots, f_p(\lambda)]$ its coordinate functions and let
- $X^T = (X_1, \ldots, X_p)$



Principal Curves and Surfaces

1: **procedure** PRINCIPAL CURVE:$(f(\lambda), X\ )$
2: **repeat**
3: $\hat{f}_j(\lambda) \leftarrow \mathbb{E}[X_j|\lambda(X) = \lambda]$, $j = 1, \ldots, p$,
4: $\hat{\lambda}_f(x) \leftarrow \arg\min_{\lambda'} \left\|x - \hat{f}(\lambda')\right\|^2$.
5: **until** convergence
6: **end procedure**



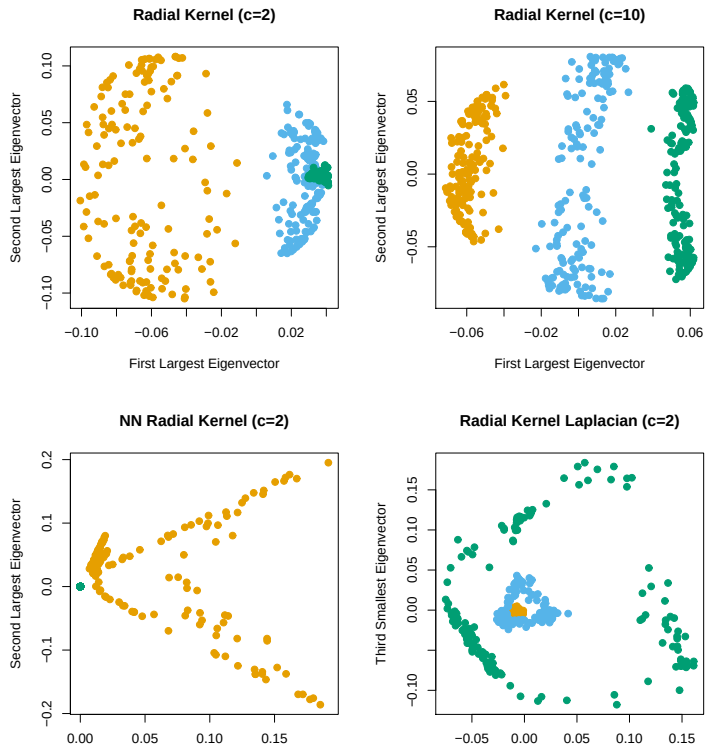
- A scatterplot smoother is used to estimate the conditional expectations in step 3: by smoothing each X_j as a function of the arc-length $\lambda(\hat{X})$.

Kernel Principal Components

- We can select any kernel function, like radial $K(x_i, x_{i'}) = e^{-\frac{\|x-x'\|^2}{c}}$.
- We set $M = 11^T/N$ and calculate double-centered version of K

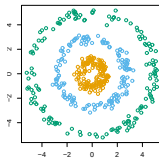
$$\tilde{K} = (I - M)K(I - M) = UD^2U^T$$

- then principal components variables are $Z = UD$.
  - The elements of the mth component z_m (mth column of Z) can be written (up to centering)
  - $z_{im} = \sum_{j=1}^{N} \alpha_{jm} K(x_i, x_j)$, where $\alpha_{jm} = \frac{u_{jm}}{d_m}$.
- Figure: Radial kernel (top) and spectral clustering without NN (bottom right) on the previous 3-'circles' example.



Spectral Clustering

- The idea is to put close points into the same cluster.
- We form a weighted adjacency graph for data samples.



Spectral Clustering

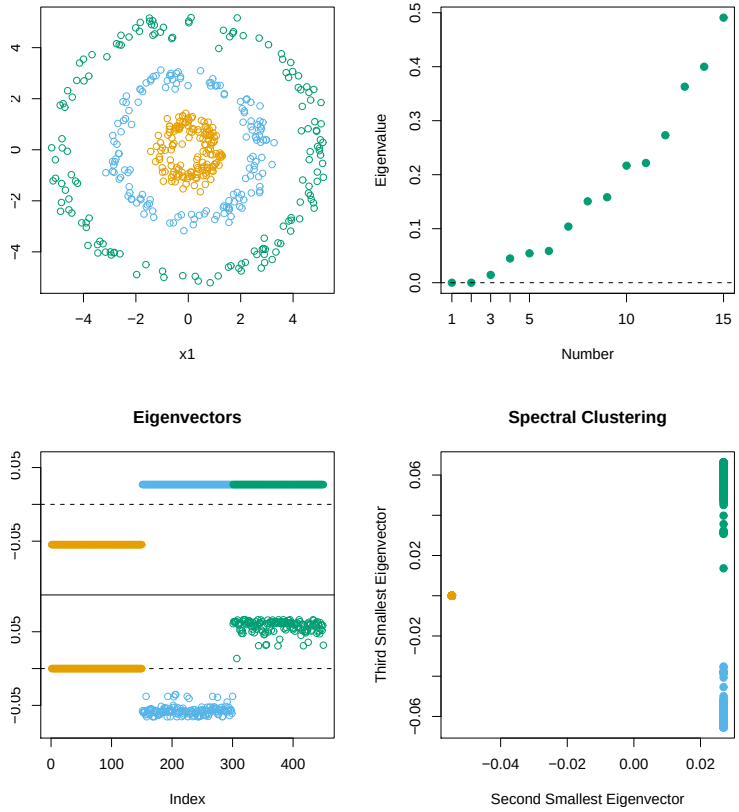
procedure SC:(X as N points in $\mathbb{R}^p$, $c > 0$ scale, $k > 0$)
    $s_{ii'} \leftarrow exp(-d(i,i')^2/c)$ # calculate the similarity matrix
    $W, G \leftarrow$ zero matrix $N \times N$
    for i, i' symmetric nearest neighbors **do**
        $w_{ii'} \leftarrow s_{ii'}$ # connect them
    end for
    for $i \in X$ **do**
        $g_{ii} \leftarrow \sum_{i'} w_{ii'}$ # the degree of vertex i
    end for
    $L \leftarrow G - W$ # the graph Laplacian (unnormalized)
    (or $\tilde{L} \leftarrow I - G^{-1}W$ # (normalized))
    find m eigenvectors $Z_{N \times m}$ with **smalest** eigenvalues of L
    return $Z_{N \times m}$ rows clustered by standard $k-means$
end procedure

Spectral Clustering

- For any vector f

$$
\begin{aligned}
f^T L f &= \sum_{i=1}^{N} g_{ii} f_i^2 - \sum_{i=1}^{N} \sum_{i'=1}^{N} f_i f_{i'} w_{ii'} \\
&= \frac{1}{2} \sum_{i=1}^{N} \sum_{i'=1}^{N} w_{ii'} (f_i - f_{i'})^2.
\end{aligned}
$$

  - $1^T L 1 = 0$ for any graph.
  - For a graph with m connected components,
    - reordered so that L is a block diagonal with a block for each component
  - then L has m eigenvectors of eigenvalue zero.
- In practice zero eigenvalues are approximated by small eigenvalues.



Separating hyperplane, Optimal separating hyperplane

- Classification, we encode the goal class by -1 and 1, respectively.
- separate the space X by a hyperplane
- **Linear Discriminant Analysis** LDA is not necessary optimal.
- **Logistic regression** finds one if it exists.
- **Perceptron** (a neural network with one neuron) finds separating hyperplane if it exists.
  - The exact position depends on initial parameters.

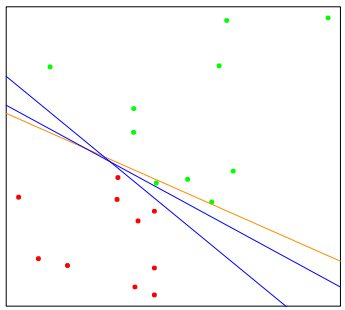


FIGURE 4.14. *A toy example with two classes separable by a hyperplane. The orange line is the least squares solution, which misclassifies one of the training points. Also shown are two blue separating hyperplanes found by the* perceptron learning algorithm *with different random starts.*

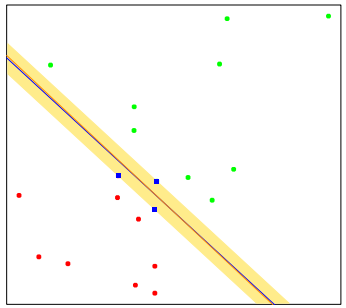


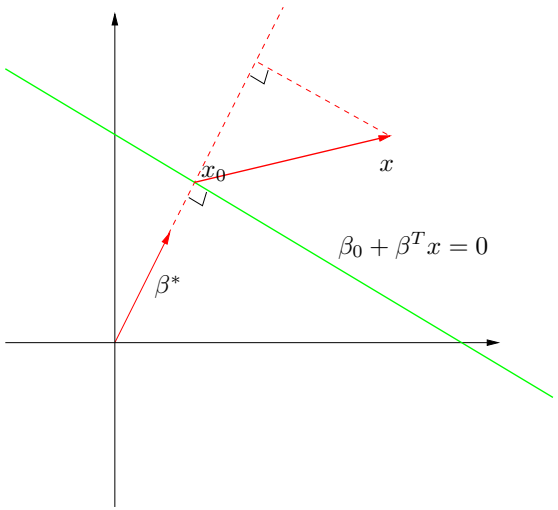
FIGURE 4.16. *The same data as in Figure 4.14. The shaded region delineates the maximum margin separating the two classes. There are three support points indicated, which lie on the boundary of the margin, and the optimal separating hyperplane (blue line) bisects the slab. Included in the figure is the boundary found using logistic regression (red line), which is very close to the optimal separating hyperplane (see Section 12.3.3).*

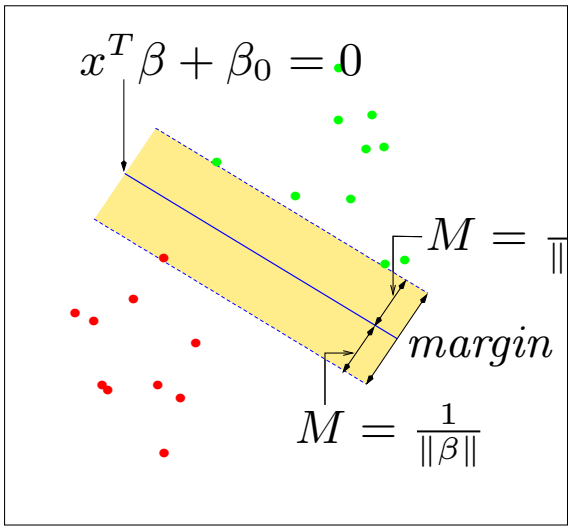
Optimal Separating Hyperplane (separble case)

We define **Optimal Separating Hyperplane** as a separating hyperplane with maximal free space M without any data point around the hyperplane.
Formally:

$$\max_{\beta,\beta_0,\|\beta\|=1} M$$

subject to $y_i(x_i^T\beta + \beta_0) \geq M$ for all $i = 1, \ldots, N$.





Formally:

$$\max_{\beta,\beta_0,\|\beta\|=1} M$$

subject to $y_i(x_i^T\beta + \beta_0) \geq M$ for all $i = 1, \ldots, N$.
We re-define: $\|\beta\| = 1$ can be moved to the condition (and redefine β_0):

$$\frac{1}{\|\beta\|} y_i(x_i^T\beta + \beta_0) \geq M$$

Since for any β and β_0 satisfying these inequalities, any positively scaled multiple satisfies them too, we can set $\|\beta\| = \frac{1}{M}$ and we get:

$$\min_{\beta,\beta_0} \frac{1}{2}\|\beta\|^2$$

subject to $y_i(x_i^T\beta + \beta_0) \geq 1$ pro $i = 1, \ldots, N$.
This is a convex optimization problem. The Lagrange function, we look for the saddle point w.r.t. β and β_0:

$$L_P = \frac{1}{2}\|\beta\|^2 - \sum_{i=1}^{N} \alpha_i[y_i(x_i^T\beta + \beta_0) - 1].$$

$$L_P = \frac{1}{2}\|\beta\|^2 - \sum_{i=1}^{N} \alpha_i [y_i(x_i^T \beta + \beta_0) - 1].$$

Setting the derivatives to zero, we obtain:

$$\beta = \sum_{i=1}^{N} \alpha_i y_i x_i$$

$$0 = \sum_{i=1}^{N} \alpha_i y_i$$

Substituing these in L_P we obtain the so–called Wolfe dual:

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{k=1}^{N} \alpha_i \alpha_k y_i y_k x_i^T x_k$$

subject to $\alpha_i \geq 0$
The solution is obtained by maximizing L_D in the positive orthant, for which standard software can be used.

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{k=1}^{N} \alpha_i \alpha_k y_i y_k x_i^T x_k$$

subject to $\alpha_i \geq 0$.
In addition the solution must satisfy the Karush–Kuhn–Tucker conditions:

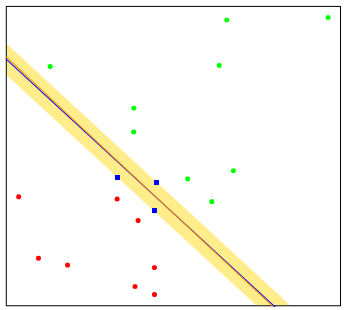
$$\alpha_i [y_i (x_i^T \beta + \beta_0) - 1] = 0$$

for any i, therefore for any $\alpha_i > 0$ must $[y_i (x_i^T \beta + \beta_0) - 1] = 0$, that means x_i is on the boundary and for all x_i outside the boundary is $\alpha_i = 0$.
The boundary is defined by x_i with $\alpha_i > 0$ – so called **support vectors**.

We classify new observations

$$\hat{G}(x) = sign(x^T \beta + \beta_0)$$

- where $\beta = \sum_{i=1}^{N} \alpha_i y_i x_i$,
- $\beta_0 = y_s - x_s^T \beta$ for any support vector $\alpha_s > 0$.



Optimal Separating Hyperplane (nonseparble case)

- We have to accept incorrectly classified instances in a non–separable case.
- We limit the number of incorrectly classified examples.

We define **slack** ξ for each data point $(\xi_1, \ldots, \xi_N) = \xi$ as follows:

- ξ_i is the distance of x_i from the boundary for x_i at the wrong side of the margin
- and $\xi_i = 0$, for x_i at the correct side.

We require $\sum_{i=1}^{N} \xi_i \leq K$.

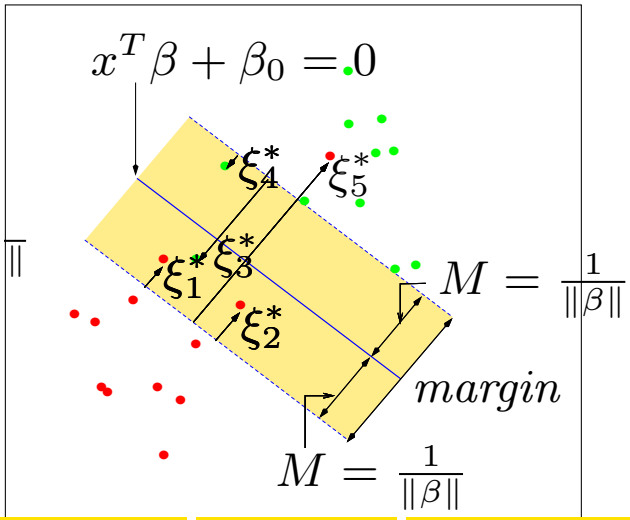
We solve the optimization problem

$$\max_{\beta, \beta_0, \|\beta\|=1} M$$

subject to:

$$y_i(x_i^T \beta + \beta_0) \geq M(1 - \xi_i)$$

where $\forall i$ is $\xi_i \geq 0$ a $\sum_{i=1}^{N} \xi_i \leq K$.



Optimal Separating Hyperplane (nonseparble case)

Again, we omit replace the condition $\|\beta\|$ by defining $M = \frac{1}{\|\beta\|}$ and optimize

$$\min \|\beta\| \text{ subject to } \begin{cases} y_i(x^T\beta + \beta_0) \geq (1 - \xi_i) \forall i \\ \xi_i \geq 0, \sum \xi_i \leq \textit{constant} \end{cases}$$

We replace the constant by a multiplicative parameter γ and solve

$$\min_{\beta, \beta_0} \frac{1}{2}\|\beta\|^2 + \gamma \sum_{i=1}^{N} \xi_i$$

subject to $\xi_i \geq 0$ and $y_i(x^T\beta + \beta_0) \geq (1 - \xi_i)$.

- We can set $\gamma = \infty$ for the separable case.
- Large γ: a complex boundary, fewer support vectors.
- Small γ: a smooth boundary, a robust model, many support vectors.
- γ usually set by crossvalidation.

We solve

$$\min_{\beta,\beta_0} \frac{1}{2}\|\beta\|^2 + \gamma \sum_{i=1}^{N} \xi_i$$

subject to $\xi_i \geq 0$ and $y_i(x_i^T\beta + \beta_0) \geq (1 - \xi_i)$.
Lagrange multipliers again for α_i, μ_i:

$$L_P = \frac{1}{2}\|\beta\|^2 + \gamma \sum_{i=1}^{N} \xi_i - \sum_{i=1}^{N} \alpha_i[y_i(x_i^T\beta + \beta_0) - (1 - \xi_i)] - \sum_{i=1}^{N} \mu_i\xi_i$$

Setting the derivative $= 0$ we get:

$$\beta = \sum_{i=1}^{N} \alpha_i y_i x_i$$

$$0 = \sum_{i=1}^{N} \alpha_i y_i$$

$$\alpha_i = \gamma - \mu_i.$$

Substitute to get Wolfe dual:

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{k=1}^{N} \alpha_i \alpha_k y_i y_k x_i^T x_k$$

and maximize L_D subject to $0 \leq \alpha_i \leq \gamma$ a $\sum_{i=1}^{N} \alpha_i y_i = 0$.
Solution satisfies:

$$\begin{aligned} \alpha_i [y_i (x_i^T \beta + \beta_0) - (1 - \xi_i)] &= 0 \\ \mu_i \xi_i &= 0 \\ \left[y_i (x_i^T \beta + \beta_0) - (1 - \xi_i) \right] &\geq 0 \end{aligned}$$

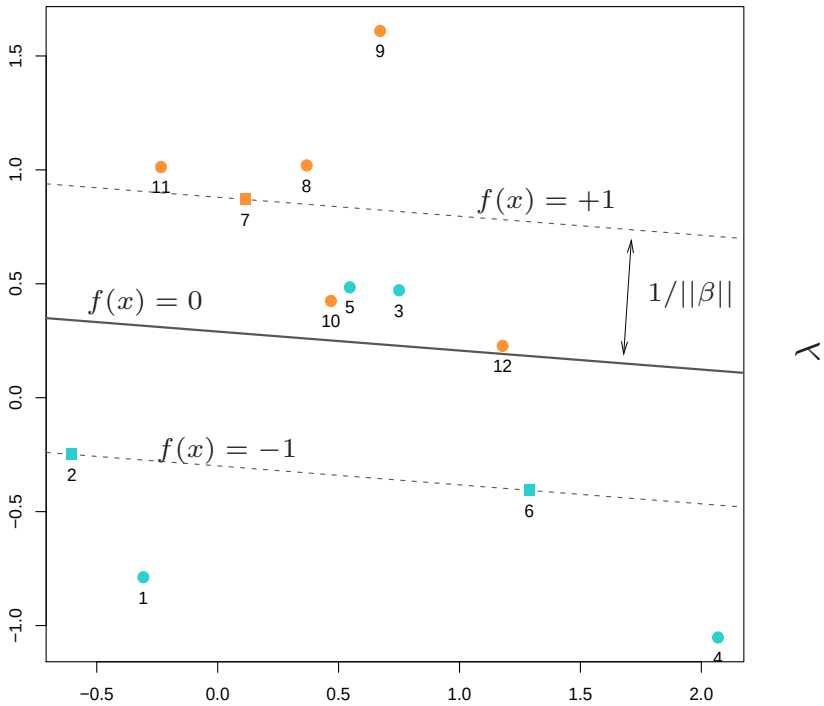
- The solution is $\hat{\beta} = \sum_{i=1}^{N} \hat{\alpha}_i y_i x_i$.
- **support points** with nonzero coefficients $\widehat{\alpha_i}$ are
  - points at the boundary
    - $\widehat{\xi_i} = 0$ (therefore $0 < \widehat{\alpha_i} < \gamma$),
  - and points on the wrong side of the margin
    - $\widehat{\xi_i} > 0$ (and $\widehat{\alpha_i} = \gamma$).
- Any point with $\widehat{\xi_i} = 0$ can be used to calculate $\widehat{\beta_0}$, typically an average.
  - $\widehat{\beta_0}$ for a boundary point $\alpha_i > 0$, $\xi_i = 0$:

$$\alpha_i \left[y_i (x^T \widehat{\beta} + \widehat{\beta_0}) - (1 - 0) \right] = 0$$

- Parameter γ settled by tuning (crossvalidation)

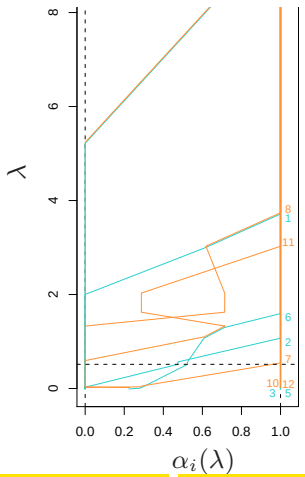
SVM Solution

- The solution is $\hat{\beta} = \sum_{i=1}^{N} \hat{\alpha}_i y_i x_i$.
- **support points** with nonzero coefficients $\widehat{\alpha}_i$ are
  - points at the boundary
    - $\widehat{\xi}_i = 0$ (therefore $0 < \widehat{\alpha}_i < \gamma$),
  - and points on the wrong side of the margin
    - $\widehat{\xi}_i > 0$ (and $\widehat{\alpha}_i = \gamma$).
- Any point with $\widehat{\xi}_i = 0$ can be used to calculate $\widehat{\beta}_0$, typically an average.



  - $\widehat{\beta}_0$ for a boundary point $\xi_i = 0$:

$$\alpha_i \left[y_i(x^T \widehat{\beta} + \widehat{\beta}_0) - (1 - 0) \right] = 0$$



- $\alpha = \xi = 0$ for points 1,4,8,9,11
- $\alpha > 0, \xi = 0$ for points 2,6,8
- missclassified points 3,5

Support Vector Machines

Let us have the training data $(x_i, y_i)_{i=1}^N$, $x_i \in \mathbb{R}^p$, y_i in $\{-1, 1\}$. We define a hyperplane

$$\{x : f(x) = x^T \beta + \beta_0 = 0\} \tag{13}$$

where $\|\beta\| = 1$.
We classify according to

$$G(x) = sign\left[x^T \beta + \beta_0\right]$$

where $f(x)$ is a signed distance of x from the hyperplane.
Support vector machines replace the scalar product $\langle x_i, x \rangle$ by a kernel function.

$$\begin{aligned}
\hat{f}(x) &= \beta x + \hat{\beta}_0 \\
\hat{f}(x) &= \sum_{k=1}^{N} \hat{\alpha}_i y_i x_i^T x + \hat{\beta}_0 \\
\hat{f}(x) &= \sum_{k=1}^{N} \hat{\alpha}_i y_i \langle x_i, x \rangle + \hat{\beta}_0 \\
\hat{f}(x) &= \sum_{k=1}^{N} \hat{\alpha}_i y_i K(x_i, x) + \hat{\beta}_0
\end{aligned}$$

SVM Example

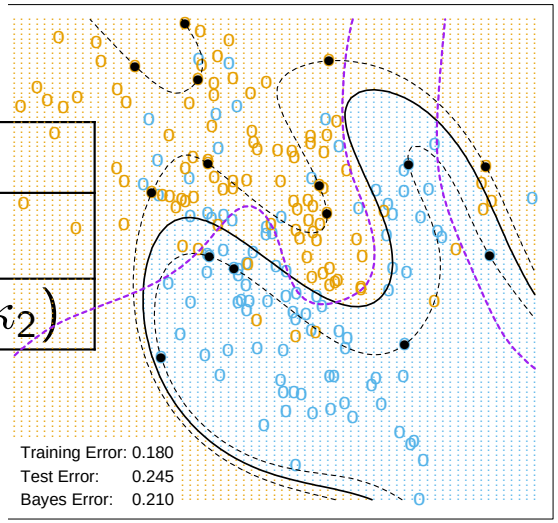
- **kernel functions** are function to replace scalar product with a scalar product in a transformed space.

dth Degree polynomial:	$K(x, x') = (1 + \langle x, x' \rangle)^d$
Radial basis	$K(x, x') = exp(\frac{-\|x-x'\|^2}{\ell})$
Neural network	$K(x, x') = tanh(\kappa_1 \langle x, x' \rangle + \kappa_2)$

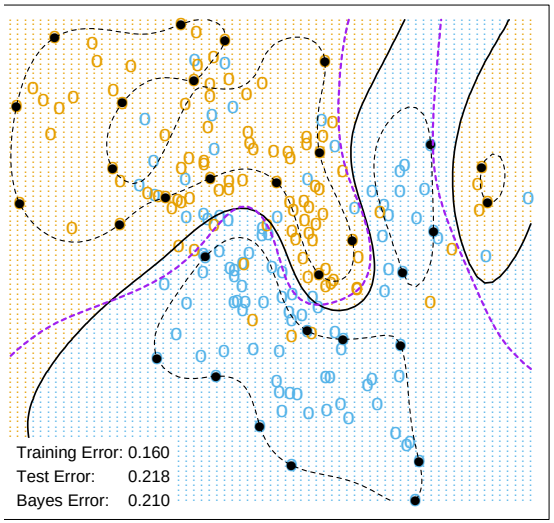
- For example a degree 2 with two dimensional input: $K(x, x') = (1 + \langle x, x' \rangle)^2 = (1 + 2x_1x_1' + 2x_2x_2' + (x_1x_1')^2 + (x_2x_2')^2 + 2x_1x_1'x_2x_2')$
- that is $M = 6$, $h_1(x) = 1$, $h_2(x) = \sqrt{2}x_1$, $h_3(x) = \sqrt{2}x_2$, $h_4(x) = x_1^2$, $h_5(x) = x_2^2$, $h_6(x) = \sqrt{2}x_1x_2$.

The classification function $\hat{f}(x) = h(x)^T\beta + \beta_0 = \sum_{i=1}^{N} \alpha_i y_i \langle h(x), h(x_i) \rangle + \beta_0$ does not need evaluation of $h(i)$, only the scalar product $\langle h(x), h(x_i) \rangle$.

SVM - Degree-4 Polynomial in Feature Space



SVM - Radial Kernel in Feature Space



String Kernels and Protein Classification

IPTSALVKETLALLSTHRTLLIANETLRIPVPVHKNHQLCTEEIFQGIGTLESQTVQGGTV
ERLFKNLSLIKKYIDGQKKKCGEERRRVNQFLDY**LQE**FLGVMNTEWI

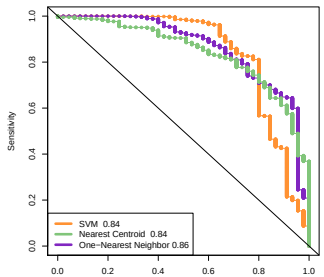
PHRRDLCSRSIWLARKIRSDLTALTESYVKHQGLWSELTEAER**LQE**NLQAYRTFHVLLA
RLLEDQQVHFTPTEGDFHQAIHTLLLQVAAFAYQIEELMILLEYKIPRNEADGMLFEKK

- Consider all possible sequences of length m.
- We define a feature map

$$\Phi_m(x) = \{\phi_a(x)\}_{a \in \mathcal{A}_m}$$

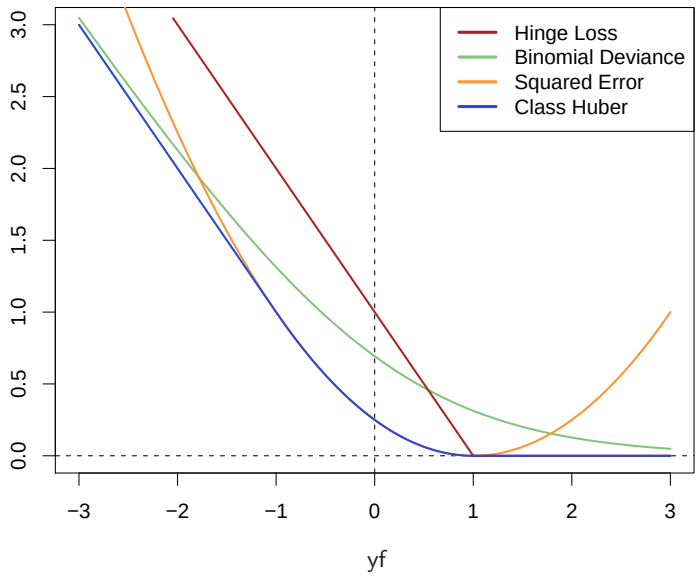
- The kernel function is the inner product:

$$K_m(x_1, x_2) = \langle \Phi_m(x_1), \Phi_m(x_2) \rangle.$$



SVM as a Penalization Method

- We fit a linear function wrt. basis $\{h_i(x)\}$: $f(x) = h^T\beta + \beta_0$.
- Consider the loss function $L(y, f) = [1 - yf]_+$
  - The optimization problem $min_{\beta_0,\beta} \sum_{i=1}^{N}[1 - yf]_+ + \lambda\|\beta\|^2$
- is equivalent to SVM
  - $\min_{\beta,\beta_0} \frac{1}{2}\|\beta\|^2 + \gamma\sum_{i=1}^{N}\xi_i$
  - subject to $\xi_i \geq 0$ and $y_i(x^T\beta + \beta_0) \geq (1 - \xi_i)$.
- is similar to smoothing splines penalty:
  - $\min_{\alpha,\alpha_0} \sum_{i=1}^{N}[1 - yf]_+ + \lambda\alpha^T\mathbf{K}\alpha$
  - where $\alpha^T\mathbf{K}\alpha = J(f)$ is the smoothing penalty.



Loss Function	$L[y, f(x)]$	Minimizing Function
Binomial Deviance	$\log[1 + e^{-yf(x)}]$	$f(x) = \log\frac{\Pr(Y=+1\|x)}{\Pr(Y=-1\|x)}$
SVM Hinge Loss	$[1 - yf(x)]_+$	$f(x) = \text{sign}[\Pr(Y=+1\|x) - \frac{1}{2}]$
Squared Error	$[y - f(x)]^2 = [1 - yf(x)]^2$	$f(x) = 2\Pr(Y=+1\|x) - 1$
"Huberised" Square Hinge Loss	$-4yf(x)$, $yf(x) < -1$ $[1 - yf(x)]_+^2$ otherwise	$f(x) = 2\Pr(Y=+1\|x) - 1$

SVM and Kernel Dimsension

- The first Simulated example
  - 100 observations of each class
  - First class: four standard normal independent features X_1, X_2, X_3, X_4.
  - Second class conditioned on $9 \leq \sum X_j^2 \leq 16$.
- Second example
  - The first one augmented with an additional six standard Gaussian noise features.
- BRUTTO: Additive spline model.
- BRUTTO and MARS has the ability to ignore noisy features.
- We can see the overfitting of SVM. The degree 2 polynomial kernel is the best since the decision boundary is quadratic.

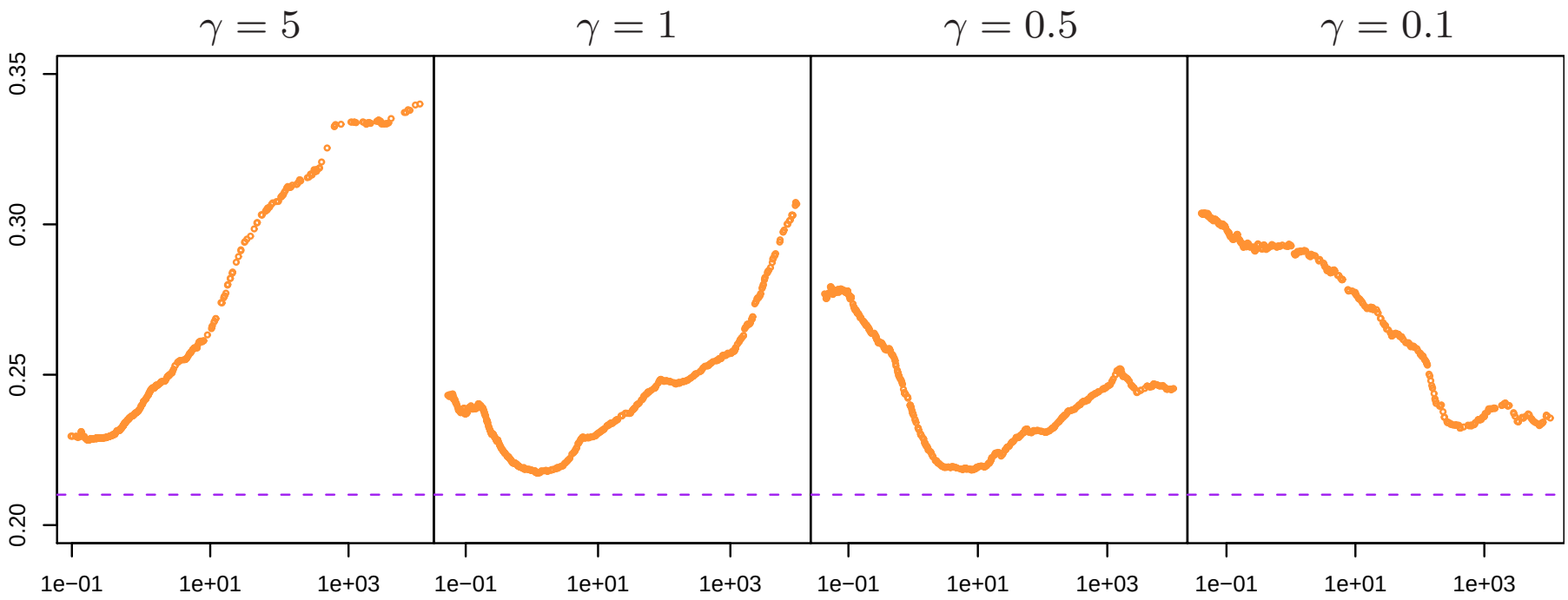
	Test Error (SE)	
Method	No Noise Features	Six Noise Features
SV Classifier	0.450 (0.003)	0.472 (0.003)
SVM/poly 2	0.078 (0.003)	0.152 (0.004)
SVM/poly 5	0.180 (0.004)	0.370 (0.004)
SVM/poly 10	0.230 (0.003)	0.434 (0.002)
BRUTO	0.084 (0.003)	0.090 (0.003)
MARS	0.156 (0.004)	0.173 (0.005)
Bayes	0.029	0.029

SVM

SVM Complexity

The SVM complexity is $m^3 + mN + mpN$, where m is the number of support vectors.

Parameter tuning for different radial basis lengthscale γ.



SVM for Regression

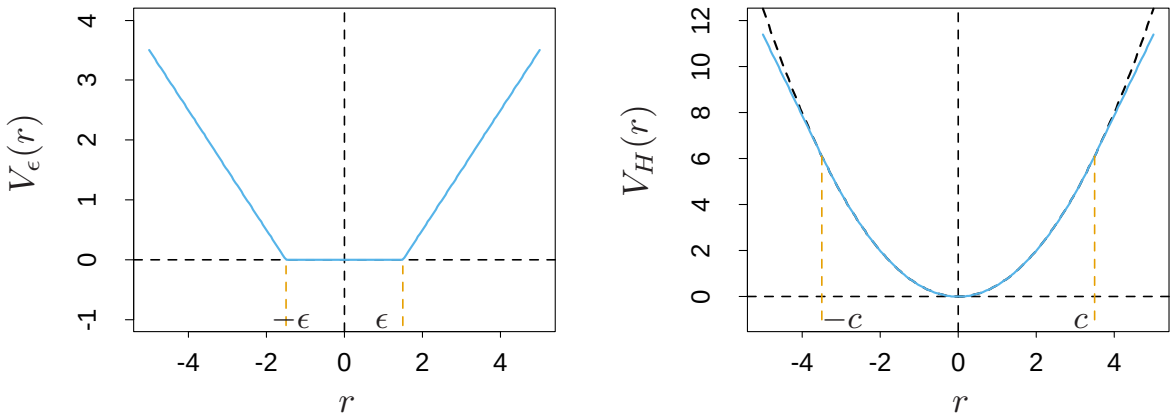
- In regression, we fit a function: $f(x) = x^T\beta + \beta_0$
- We consider error function V_ϵ (left figure)

$$V_\epsilon(r) = \begin{cases} 0 & \text{if } |r| < \epsilon, \\ |r| - \epsilon, & \text{otherwise.} \end{cases}$$

- and minimize:

$$H(\beta, \beta_0) = \sum_{i=1}^{N} V_\epsilon(y_i - f(x_i)) + \frac{\lambda}{2}\|\beta\|^2$$

-



SVM for Regression 2

- The solution has the form: $\hat{\alpha}_i, \hat{\alpha}_i^* \geq 0$

$$\begin{aligned} \hat{\beta} &= \sum_{i=1}^{N} (\hat{\alpha}_i^* - \hat{\alpha}_i) x_i, \\ \hat{f}(x) &= \sum_{i=1}^{N} (\hat{\alpha}_i^* - \hat{\alpha}_i) \langle x, x_i \rangle + \beta_0, \end{aligned}$$

- and solve the quadratic programming problem

$$\min_{\alpha_i, \alpha_i^*} \epsilon \sum_{i=1}^{N} (\alpha_i^* + \alpha_i) - \sum_{i=1}^{N} y_i (\alpha_i^* - \alpha_i) + \frac{1}{2} \sum_{i,i'=1}^{N} (\alpha_i^* - \alpha_i)(\alpha_{i'}^* - \alpha_{i'}) \langle x_i, x_{i'} \rangle$$

- subject to the constraints

$$\begin{aligned} 0 \leq \alpha_i,\ \alpha_i^* \leq 1/\lambda, \\ \sum_{i=1}^{N} (\alpha_i^* - \alpha_i) = 0, \\ \alpha_i \alpha_i^* = 0. \end{aligned}$$

- Support vectors are those with nonzero $(\hat{\alpha}_i^* - \hat{\alpha}_i)$.
- With scaled response y, you may use the default ϵ.
- λ is tuned by cross-validation.

SVR

sklearn.svm.SVR

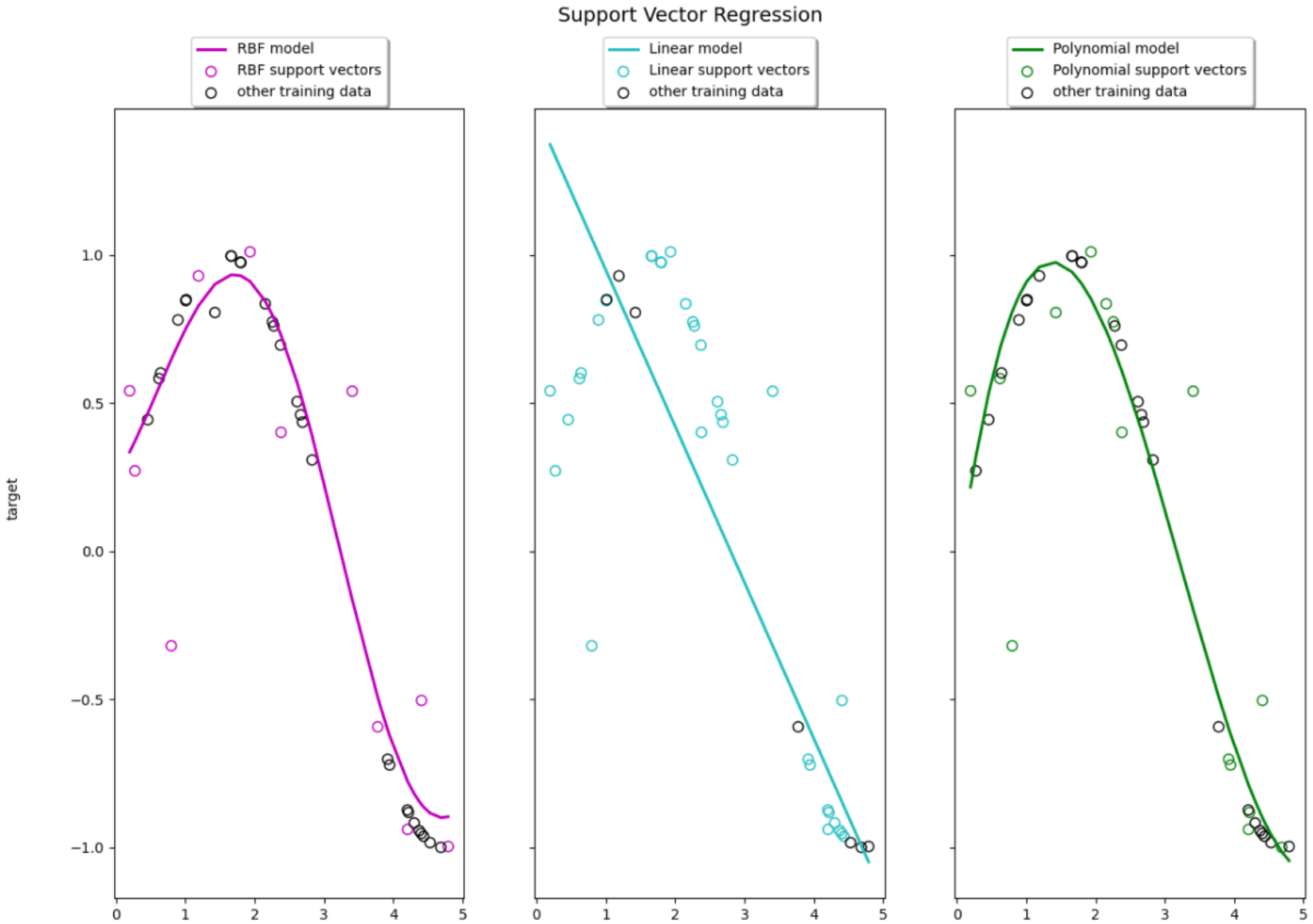


Table of Contens