Kernel-based methods

4 Apr 2023

Table of contents

Kernel-based methods

- depend only on some *distance function* induced between pairs of points
  - potentially a high (even infinite-dimensional!) space
  - mapping original values to high dim (e.g. high-order interactions)
- kernel function k
  - classification: $\hat{y}_i(\mathbf{x}') = \text{sign} \sum w_i y_i k(\mathbf{x}_i, \mathbf{x}')$
  - regression: (the same but without the sign()!)
- for further efficiency can also use **low-rank approximations** of $k()$

Kernel smoothers

- kernel density estimation
- Nadaraya-Watson kernel regession

Separating hyperplanes

- ESL section 4.5
- Regress $\mathbf{y} \in \{-1, 1\}$ on $\mathbf{x}$: solve for $\mathbf{X}\beta = 0$ (write as $\beta_0 + \beta^\top \mathbf{x} = 0$, i.e. separate intercept)
- (equivalent to linear discriminant analysis)
- Rosenblatt's algorithm
  - $(\mathbf{X}\beta)/||\beta||$ is the signed distance to the separating plane
  - minimize $-\sum_{i \in M} y_i(\mathbf{X}\beta)$ (sum of misclassified distances)
    - gradient wrt $\beta = -\sum(y_i x_i)$
  - stochastic gradient descent (pointwise): adjust β by $\rho \mathbf{y}_i X_i$ at each step
- elegant but not practical (non-unique, slow, non-convergent if not separable)
- → penalized version in a larger basis space
- argmin(β) of $\frac{1}{2}||\beta||^2$ subject to $y_i(\mathbf{X}\beta) \geq 1$

- “standard” convex optimization problem

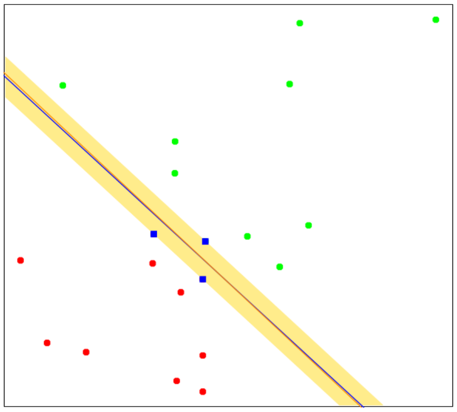


FIGURE 4.16. *The same data as in Figure 4.14. The shaded region delineates the maximum margin separating the two classes. There are three support points indicated, which lie on the boundary of the margin, and the optimal separating hyperplane (blue line) bisects the slab. Included in the figure is the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

support vector machines for the non-separable case

- ESL chapter 12
- $y_i(X_i\beta \geq M(1-\xi_i)$
- linear loss function on misclassification distances + L2 penalty
- or $\min\frac{1}{2}||\beta||^2 + C\sum\xi_i$
- C is the hyperparameter
- quadratic programming problem

SVMs and kernels (ESL 12.3)

- alternative formulation

$$\begin{aligned} f(x_i) &= X_i^\top \beta + \beta_0 \\ &= \sum \alpha_j y_j \left\langle h(x_i), h(x_j) \right\rangle + \beta_0 \end{aligned}$$

where α_i is a different parameterization

- $\langle h(.), h(.) \rangle$ is a **kernel function**
- linear SVM finds a separating hyperplane based on distances
- polynomial distance: $(1 + \langle x_i, x_j \rangle)^d$
- polynomial d for n inputs (plus intercept) gives rise to a $C(n+2, d)$-dimensional space
- **radial basis function** $\exp\left(-\gamma ||x_i - x_j||^2\right)$
  - infinite-dimensional (think of Taylor expansion)
  - **length scale** $1/\gamma$

SVMs for regression

- fits a loss function $\max(0, |r| - \epsilon)$

kernels

- "kernel trick" works very generally, but only for L2 penalty
- ESL 12.3.7: cost of optimizing via kernel is $O(N^2)$ not $O(MN^2)$ (where N is number of training points, M is dimension of the feature space)

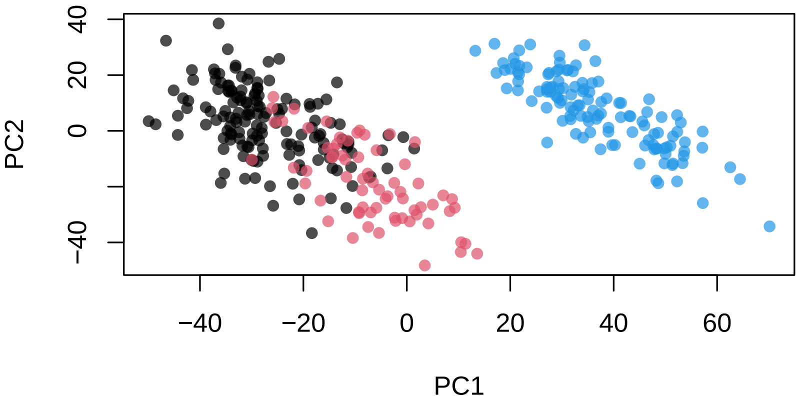
kernel PCA

- Schölkopf, Smola, and Müller (1997)
- we can do PCA by SVD (works if $p > n$): complexity is $O(\min(np^2, n^2p)$ https://mathoverflow.net/a/221216 (`stats:::prcomp.default`)
- or by computing covariance and then computing eigenvectors (only works for $n < p$): On^3
- kPCA: map $\Phi : \mathbf{R}^n \to F$
- find $K = \langle \Phi(x_i)\Phi(x_j) \rangle$
  - never worse than n^3, no matter how big the **feature space** is (even infinite)
  - better than linear PCA if $p > n$

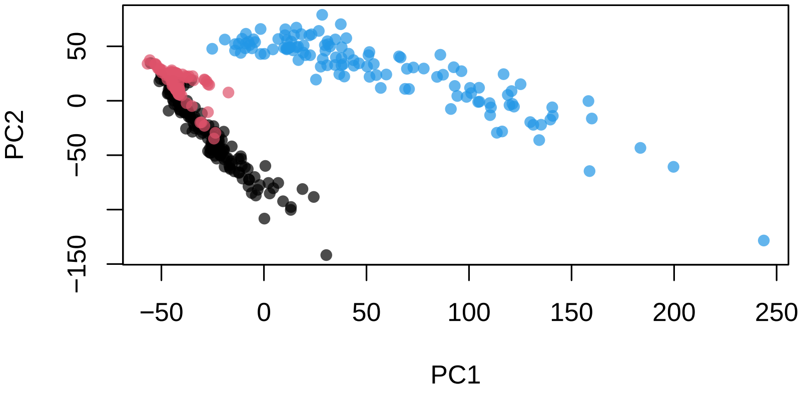
Schölkopf, Bernhard, Alexander Smola, and Klaus-Robert Müller. 1997. "Kernel Principal Component Analysis." In *Artificial Neural Networks — ICANN'97*, edited by Wulfram Gerstner, Alain Germond, Martin Hasler, and Jean-Daniel Nicoud, 583–88. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer. https://doi.org/10.1007/BFb0020217.

kpca example

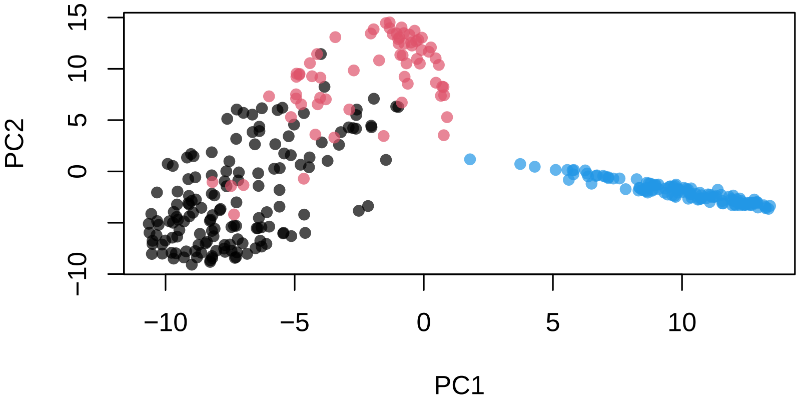
```
library(palmerpenguins)
library(tidyverse)
library(kernlab)
pX <- (penguins
    |> select(where(is.numeric))
    |> select(-year)
    |> as.matrix()
    |> scale()
    |> na.omit()
)
## want species to match with NA-adjusted matrix!
ss <- penguins |> drop_na(where(is.numeric)) |> pull(species)
cc <- adjustcolor(palette()[c(1,2,4)], alpha.f = 0.7)
pfun <- function(k) {
    plot(rotated(k),col= cc[ss], pch = 16,
         xlab = "PC1", ylab = "PC2")
}
k0 <- kpca(pX, kernel = "vanilladot", kpar = list())
pfun(k0)
```



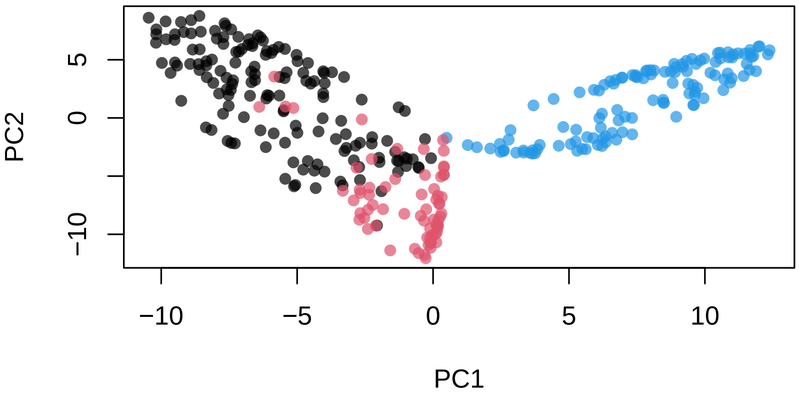
```
k1 <- kpca(pX, kernel = "polydot", kpar = list(degree = 2))
pfun(k1)
```



```
k2 <- kpca(pX, kernel = "rbfdot", kpar = list(sigma = 0.5))
pfun(k2)
```

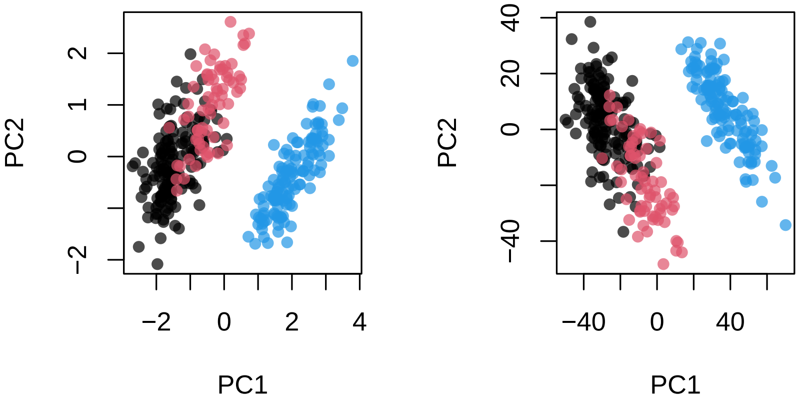


```
k3 <- kpca(pX, kernel = "rbfdot", kpar = list(sigma = 1))
pfun(k3)
```

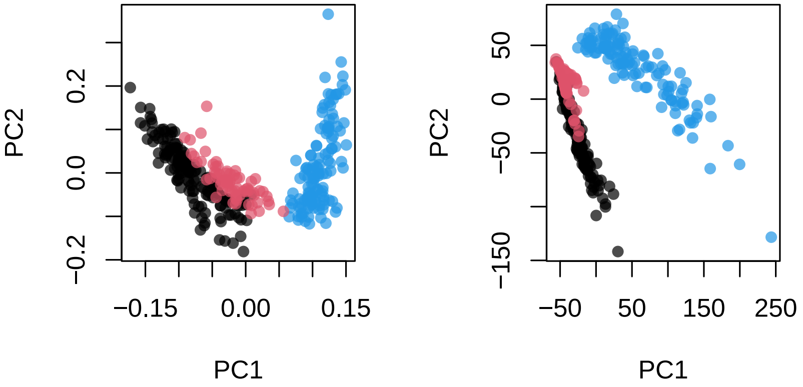


expanded PCA by hand

```r
pfun2 <- function(p) {
    plot(p$scores[,1], p$scores[,2], col= cc[ss], pch = 16,
         xlab = "PC1", ylab = "PC2")
}
p0 <- princomp( ~ ., data = as.data.frame(pX))
polyform <- (sprintf("poly(%s, degree = 2)", paste(colnames(pX), collapse = ","))
    |> reformulate()
)
X <- model.matrix(polyform, data = as.data.frame(pX))
p1 <- princomp( polyform, data = as.data.frame(pX))
par(mfrow = c(1, 2)); pfun2(p0); pfun(k0)
```



```
par(mfrow = c(1, 2)); pfun2(p1); pfun(k1)
```



how should one choose a kernel? - if kPCA is a step in a pipeline, make the kernel type and parameters (e.g. scale for RBF) part of the tuning process

Gaussian processes

- Rasmussen and Williams (2005); Krasser (2018); Krasser (2020)
- motivated by Bayesian context, or from classical **geostatistics** (kriging)
- interpolation vs. approximation
- "Under the assumption of Gaussian observation noise the computations needed to make predictions are tractable and are dominated by the inversion of a n × n matrix."
- zero-mean Gaussian prior: $\mathbf{w} \sim N(0, \Sigma_p)$
- Σ_p? **positive definite** function of distance
    - $\mathbf{x}^\top \Sigma \mathbf{x} > 0$ for all $\mathbf{x} \neq 0$
    - all eigenvalues of Σ are positive
- only certain **autocovariance functions** $f(r)$ satisfy this condition for all possible $\mathbf{x}$: RBF, Matérn, …

Rasmussen, Carl Edward, and Christopher K. I. Williams. 2005. *Gaussian Processes for Machine Learning*. Cambridge, Mass: The MIT Press.

Krasser, Martin. 2018. "Gaussian Processes." http://krasserm.github.io/2018/03/19/gaussian-processes/.

———. 2020. "Sparse Gaussian Processes." http://krasserm.github.io/2020/12/12/gaussian-processes-sparse/.

GP prior

$$\begin{pmatrix} f \\ f_* \end{pmatrix} \sim N\left(0, \begin{pmatrix} \mathbf{K} & \mathbf{K}_* \\ \mathbf{K}_*^\top & \mathbf{K}_{**} \end{pmatrix}\right)$$

conditional posterior distribution

$$\begin{aligned} \mu_* &= \mathbf{K}_*^\top \mathbf{K}^{-1} f \\ \Sigma_* &= \mathbf{K}_{**} - \mathbf{K}_*^\top \mathbf{K}^{-1} \mathbf{K} \end{aligned}$$

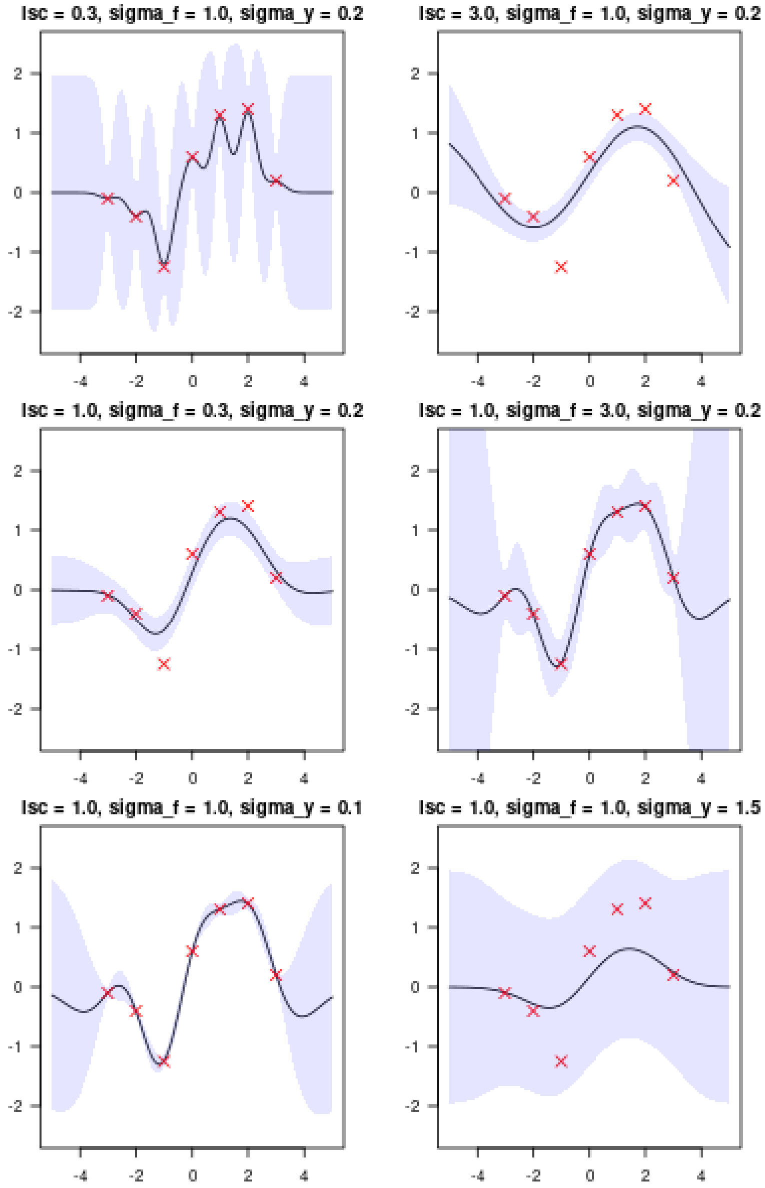
kriging equations

observation variance/measurement error

- **nugget variance**
- allow smoothing rather than interpolation
- cf SVM without complete separation

hyperparameters

- variance of Gaussian process
- observation variance ‘scale’ (residual variance)
  - *could* estimate by cross-validation etc.
  - but have MLE, Bayesian estimates immediately available …
  - level of uncertainty between interpolating points
  - can estimate directly from MVN MLE as $y^\top C y$
- length scale

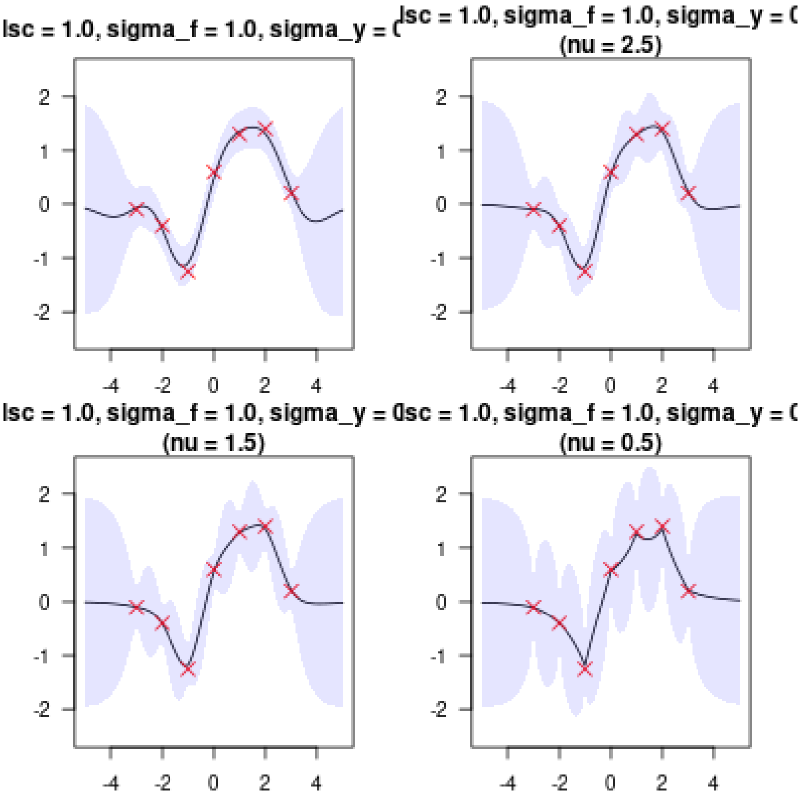


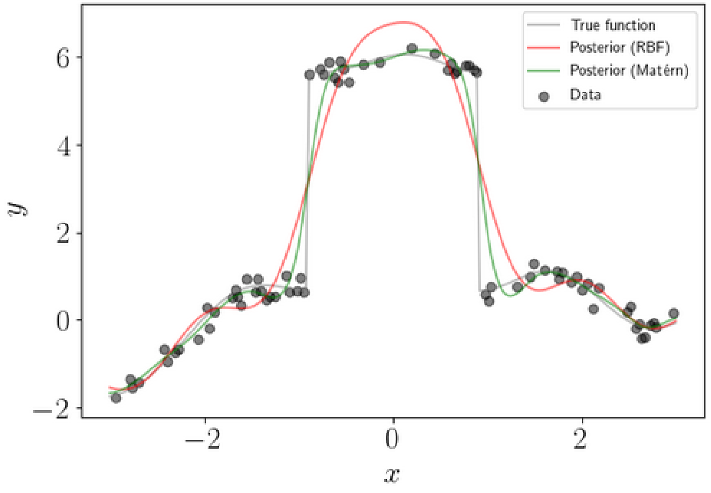
kernel shape

- **smoothness**
- RBF is infinitely differentiable
- Matérn functions

  - $\nu = 1/2$, $\exp(-|d|)$
  - $\nu = 3/2$, $(1 + \sqrt{3}d)\exp(-\sqrt{3}d)$
  - $\nu = 5/2$, $(1 + \sqrt{5}d + 5/3d^2)\exp(-\sqrt{5}d)$

- otherwise Bessel function K_ν (ugh)





Jones (2021)

Jones, Andy. 2021. "The Matérn Class of Covariance Functions." *Andy Jones*. https://andrewcharlesjones.github.io/journal/matern-kernels.html.

stationarity and isotropy

- **isotropy**: exchangeability of parameters/directions
  - **separable** priors (different variance for each direction)
  - could?? model prior correlation as well)
- **stationarity**: kernel depends only on $||x_i - x_j||$
  - non-stationary: location-dependent, center at mean?
  - or model linear covariates + GP

Paciorek and Schervish (2003)

Paciorek, Christopher, and Mark Schervish. 2003. "Nonstationary Covariance Functions for Gaussian Process Regression." In *Advances in Neural Information Processing Systems*. Vol. 16. MIT Press. https://proceedings.neurips.cc/paper/2003/hash/326a8c055c0d04f5b06544665d8bb3ea-Abstract.html.

GPs and splines

- ; Rasmussen and Williams (2005)
- `?mgcv::smooth.construct.gp.smooth.spec`
- Kammann and Wand (2003)
  - choose Matérn with $\nu = 3/2$

Rasmussen, Carl Edward, and Christopher K. I. Williams. 2005. *Gaussian Processes for Machine Learning*. Cambridge, Mass: The MIT Press.

Kammann, E. E., and M. P. Wand. 2003. "Geoadditive Models." *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 52 (1): 1–18. https://doi.org/10.1111/1467-9876.00385.

  - choose scale == max distance between sample points
  - only need to estimate σ_f^2
  - fit linear mixed model

$$
\begin{aligned}
\mathbf{Z} &= K(||x_i - \kappa_k||/\rho) \\
\Omega &= K(||\kappa_k - \kappa_k'||/\rho) \\
\mathbf{y} &= \mathbf{X}\beta + \mathbf{Z}u + \epsilon
\end{aligned}
$$

minimize $||\mathbf{y} - \mathbf{X}\beta + \mathbf{Z}||_2^2 + ||\Omega^{-1}u||_2^2$

Bayesian model tuning/hyperparameter tuning

- `tune_bayes`
- https://www.tidymodels.org/learn/work/bayes-opt/

sparsity: nearest-neighbour

combining kernels

packages

`plgp`, `spectralGP`, `GauPro`, `tfprobability`, `mlegp`, …

sparse GPs: `psgp`