ipop.r

denis

2021-07-12

#!/usr/bin/r  
  
# However the data are represented, whether in an array or a network, the  
# analysis of the data is often facilitated by using “association” matrices. The  
# most familiar type of association matrix is perhaps a correlation matrix. We  
# will encounter and use other types of association matrices in Chapter 8.  
  
# In this chapter we discuss a wide range of basic topics related to vectors   
# of real  
# numbers. Some of the properties carry over to vectors over other fields, such  
# as complex numbers, but the reader should not assume this. Occasionally, for  
# emphasis, we will refer to “real” vectors or “real” vector spaces, but unless   
# it  
# is stated otherwise, we are assuming the vectors and vector spaces are real.  
# The topics and the properties of vectors and vector spaces that we emphasize  
# are motivated by applications in the data sciences.  
real <- vector(mode = "logical", length = 0L)  
real

## logical(0)

# Abstract  
# kernlab is an extensible package for kernel-based machine learning methods   
# in R. It takes  
# advantage of R’s new S4 object model and provides a framework for creating   
# and using kernel-  
# based algorithms. The package contains dot product primitives (kernels),   
# implementations  
# of support vector machines and the relevance vector machine, Gaussian   
# processes, a ranking  
# algorithm, kernel PCA, kernel CCA, kernel feature analysis, online kernel   
# methods and a  
# spectral clustering algorithm. Moreover it provides a general purpose   
# quadratic programming  
# solver, and an incomplete Chomsky decomposition method.  
# Keywords: kernel methods, support vector machines, quadratic programming,   
# ranking, clustering,  
# S4, R.  
n = 1  
S4 <- rnorm(n, mean = 0, sd = 1)  
r = 1  
CCA <- kernel(coef = "daniell", m = 2, r, name = "unknown")  
S4

## [1] -0.9086617

CCA

## Daniell(2)   
## coef[-2] = 0.2  
## coef[-1] = 0.2  
## coef[ 0] = 0.2  
## coef[ 1] = 0.2  
## coef[ 2] = 0.2

# 1. Introduction  
# Machine learning is all about extracting structure from data, but it is often   
# difficult to solve prob-  
# lets like classification, regression and clustering in the space in which the   
# underlying observations  
# have been made.  
# Kernel-based learning methods use an implicit mapping of the input data into   
# a high dimensional  
# feature space defined by a kernel function, i.e., a function returning the   
# inner product hΦ(x), Φ(y)i  
# between the images of two data points x, y in the feature space. The learning   
# then takes place  
# in the feature space, provided the learning algorithm can be entirely   
# rewritten so that the data  
# points only appear inside dot products with other points. This is often   
# referred to as the “kernel  
# trick” (Schölkopf and Scold 2002). More precisely, if a projection   
# V : X → H is used, the dot  
# product hp(x), V(y)i can be represented by a kernel function k  
# k(x, y) = hp(x), V(y)i,  
k <- function(hp){  
 i <- 0  
 hp <- c(list(i), 0, i)  
  
   
 c(hp, i)  
}  
k(hp)

## [[1]]  
## [1] 0  
##   
## [[2]]  
## [1] 0  
##   
## [[3]]  
## [1] 0  
##   
## [[4]]  
## [1] 0

# which is computationally simpler than explicitly projecting x and y into the   
# feature space H.  
# One interesting property of kernel-based systems is that, once a valid kernel   
# function has been  
# selected, one can practically work in spaces of any dimension without paying   
# any computational  
# cost, since feature mapping is never effectively performed. In fact, one does   
# not even need to know  
# which features are being used.  
# Another advantage is the that one can design and use a kernel for a particular   
# problem that could be  
# applied directly to the data without the need for a feature extraction   
# process. This is particularly  
# important in problems where a lot of structure of the data is lost by the   
# feature extraction process  
# (e.g., text processing). The inherent popularity of kernel-based learning   
# methods allows one to  
# use any valid kernel on a kernel-based algorithm.  
feature <- c(10, type = c("O", "I", "F", "M", "2"))  
feature

## type1 type2 type3 type4 type5   
## "10" "O" "I" "F" "M" "2"

# 1.1. Software review  
# The most prominent kernel based learning algorithm is without doubt the   
# support vector machine2  
# kernlab – An S4 Package for Kernel Methods in R  
# (SVM), so the existence of many support vector machine packages comes as   
# little surprise. Most  
# of the existing SVM software is written in C or C++, e.g. the award winning   
# libsvm 1 (Chang and  
   
# Lin 2001), Sunlight 2 (Joachims 1999), SVMTorch 3 , Royal Holloway Support   
# Vector Machines 4 ,  
# myS 5 , and M-SVM 6 with many packages providing interfaces to   
# MATLAB (such as libsvm),  
# and even some native MATLAB toolboxes 7 8 9 .  
# Putting SVM specific software aside and considering the abundance of other   
# kernel-based ago-  
# rhythms published nowadays, there is little software available implementing   
# a wider range of kernel  
# methods with some exceptions like the Spider 10 software which provides   
# a MATLAB interface to  
# various C/C++ SVM libraries and MATLAB implementations of various   
# kernel-based algorithms,  
# Torch 11 which also includes more traditional machine learning algorithms,   
# and the occasional  
# MATLAB or C program found on a personal web page where an author includes   
# code from a  
# published paper.  
kernlab::.\_\_C\_\_ipop

## Class "ipop" [package "kernlab"]  
##   
## Slots:  
##   
## Name: primal dual how  
## Class: vector numeric character