MAP 553 - Statistical learning and nonparametric estimation
Numerical project
Report

EZZAKI Mahmoud   ROHRMANN Till

December 28, 2012

1 Problem description

2 Data exploration

3 Interpretation

A Annex

```
# q.R
#
#This file creates the graphs for our problem. It calculates the regression
# function with the GDP per capita as its dependent variable for different
# independent variables:
#   - Export ratio to GDP per capita
#   - Import ratio to GDP per capita
#   - Worker ratio to total population
#   - Life expectation
#   - Average school years
#   - Political instability
#   - Political rights
#   - Civil liberties
#
# Moreover, the following 2 dimensional regression functions are calculated with
# the GDP per capita as its dependent variable:
#   - Export ratio to GDP per capita and import ratio to GDP per capita
#   - Export ratio to GDP per capita and life expectation
#   - Export ratio to GDP per capita and worker ratio to total population
#   - Political rights and civil liberties

source("normalize.R")
source("gaussianNoyau.R")
source("rectangularNoyau.R")
source("localPolynomsEstimation.R")
source("leastSquaresEstimation.R")
source("plotHelper1D.R")
source("plotHelper2D.R")
source("plotHelperD1D.R")
source("trigonometricDictionary.R")
source("derive1D.R")
source("transformEstimator1D.R")

data <- read.csv("../donn es/BARLEE/base.csv")

cols <- c(3:7, 9:13, 15:29, 31:40, 55:64, 41:48)
workdata <- data[, cols]
GDP <- c(workdata$GDP.65, workdata$GDP.70, workdata$GDP.75, workdata$GDP.80, workdata$GDP.85)
Worker <- c(workdata$Worker.65, workdata$Worker.70, workdata$Worker.75, workdata$Worker.80,
    workdata$Worker.85)
Life.expectation <- c(workdata$Life.expectation.65, workdata$Life.expectation.70,
    workdata$Life.expectation.75, workdata$Life.expectation.80, workdata$Life.expectation.85)
Average.school.years <- c(workdata$Average.school.years.65, workdata$Average.school.years.70,
    workdata$Average.school.years.75, workdata$Average.school.years.80, workdata$Average.school.years.85)
Export <- c(workdata$Export.60.65, workdata$Export.65.70, workdata$Export.70.75,
    workdata$Export.75.80, workdata$Export.80.85)
Import <- c(workdata$Import.60.65, workdata$Import.65.70, workdata$Import.70.75,
    workdata$Import.75.80, workdata$Import.80.85)
PoInst <- c(workdata$Political.Instability.60.65, workdata$Political.Instability.65.70,
    workdata$Political.Instability.70.75, workdata$Political.Instability.75.80,
            workdata$Political.Instability.80.85)

PoRights <- c(workdata$Political.rights.70.75, workdata$Political.rights.75.80,
              workdata$Political.rights.80.85, workdata$Political.rights.85.90)
CivLibs <- c(workdata$Civil.Liberties.70.75, workdata$Civil.Liberties.75.80, workdata$Civil.Liberties.80.85,
    workdata$Civil.Liberties.85.90)
GDPRights <- c(workdata$GDP.70, workdata$GDP.75, workdata$GDP.80, workdata$GDP.85)

domainGDP = c(min(GDP, na.rm = TRUE), max(GDP, na.rm = TRUE))
GDP <- t(normalize(GDP))

domainWorker = c(min(Worker, na.rm = TRUE), max(Worker, na.rm = TRUE))
Worker <- t(normalize(Worker))

domainLifeExpectation = c(min(Life.expectation, na.rm = TRUE), max(Life.expectation,
    na.rm = TRUE))
Life.expectation <- t(normalize(Life.expectation))

domainAvgSchoolYears = c(min(Average.school.years, na.rm = TRUE), max(Average.school.years,
    na.rm = TRUE))
Average.school.years <- t(normalize(Average.school.years))

domainExport <- c(min(Export, na.rm = TRUE), max(Export, na.rm = TRUE))
Export <- t(normalize(Export))

domainImport <- c(min(Import, na.rm = TRUE), max(Import, na.rm = TRUE))
Import <- t(normalize(Import))

domainPoInst <- c(min(PoInst, na.rm = TRUE), max(PoInst, na.rm = TRUE))
PoInst <- t(normalize(PoInst))

domainCivLibs <- c(min(CivLibs, na.rm = TRUE), max(CivLibs, na.rm = TRUE))
CivLibs <- t(normalize(CivLibs))

domainPoRights <- c(min(PoRights, na.rm = TRUE), max(PoRights, na.rm = TRUE))
```

```r
PoRights <- t(normalize(PoRights))

domainGDPRights <- c(min(GDPRights, na.rm = TRUE), max(GDPRights, na.rm = TRUE))
GDPRights <- t(normalize(GDPRights))

final <- data.frame(GDP = GDP, Worker = Worker, Life.expectation = Life.expectation,
    Average.school.years = Average.school.years, Export = Export, Import = Import)

dict <- trigonometricDictionary(dimension, 0, 1)
noyau <- gaussianNoyau(dimension)

ordre <- 3
dimension <- 1
blocksize <- 10
h <- 0.01

final <- data.frame(GDP = GDP, Export = Export)
final <- final[complete.cases(final), ]

estimatorGDPExp <- localPolynomsEstimation(final$Export, final$GDP, ordre, dimension,
    noyau, blocksize, 0.005, 1, 0.005)
plotHelper1D(final$Export, domainExport, "Export ratio", final$GDP, domainGDP, "GDP",
    estimatorGDPExp, "Export ratio to GDP")

estimatorGDPExpLS <- leastSquaresEstimation(final$Export, final$GDP, dimension, dict,
    blocksize, 1, 20)
plotHelper1D(final$Export, domainExport, "Export ratio", final$GDP, domainGDP, "GDP",
    estimatorGDPExpLS, "Export ratio to GDP LS")

estimatorGDPExpD <- derive1D(estimatorGDPExp, h)
plotHelperD1D(domainExport, "Export ratio", domainGDP, "Derived GDP", estimatorGDPExpD,
    "Derivation export ratio to GDP")

final <- data.frame(Import = Import, GDP = GDP)
final <- final[complete.cases(final), ]

estimatorGDPImp <- localPolynomsEstimation(final$Import, final$GDP, ordre, dimension,
    noyau, blocksize, 0.005, 1, 0.005)
plotHelper1D(final$Import, domainImport, "Import ratio", final$GDP, domainGDP, "GDP",
    estimatorGDPImp, "Import ratio to GDP")

estimatorGDPImpLS <- leastSquaresEstimation(final$Import, final$GDP, dimension, dict,
    blocksize, 1, 20)
plotHelper1D(final$Import, domainImport, "Import ratio", final$GDP, domainGDP, "GDP",
    estimatorGDPImpLS, "Import ratio to GDP LS")

estimatorGDPImpD <- derive1D(estimatorGDPImp, h)
plotHelperD1D(domainImport, "Import ratio", domainGDP, "Derived GDP", estimatorGDPImpD,
    "Derivation import ratio to GDP")

final <- data.frame(GDP = GDP, Life.expectation = Life.expectation)
final <- final[complete.cases(final), ]

estimatorGDPLife <- localPolynomsEstimation(final$Life.expectation, final$GDP, ordre,
    dimension, noyau, blocksize, 0.005, 1, 0.005)
plotHelper1D(final$Life.expectation, domainLifeExpectation, "Life expectation", final$GDP,
    domainGDP, "GDP", estimatorGDPLife, "Life expectation to GDP")

estimatorGDPLifeLS <- leastSquaresEstimation(final$Life.expectation, final$GDP, dimension,
    dict, blocksize, 1, 20)
plotHelper1D(final$Life.expectation, domainLifeExpectation, "Life expectation", final$GDP,
    domainGDP, "GDP", estimatorGDPLifeLS, "Life expectation to GDP LS")

estimatorGDPLifeD <- derive1D(estimatorGDPLife, h)
plotHelperD1D(domainLifeExpectation, "Life expectation", domainGDP, "Derived GDP",
    estimatorGDPLifeD, "Derivation life expectation to GDP")

final <- data.frame(GDP = GDP, Worker = Worker)
final <- final[complete.cases(final), ]

estimatorGDPWorker <- localPolynomsEstimation(final$Worker, final$GDP, ordre, dimension,
    noyau, blocksize, 0.005, 1, 0.005)
plotHelper1D(final$Worker, domainWorker, "Worker ratio to population", final$GDP,
    domainGDP, "GDP", estimatorGDPWorker, "Worker ratio to GDP")

estimatorGDPWorkerLS <- leastSquaresEstimation(final$Worker, final$GDP, dimension,
    dict, blocksize, 1, 20)
plotHelper1D(final$Worker, domainWorker, "Worker ratio to population", final$GDP,
    domainGDP, "GDP", estimatorGDPWorkerLS, "Worker ratio to GDP LS")

estimatorGDPWorkerD <- derive1D(estimatorGDPWorker, h)
plotHelperD1D(domainWorker, "Worker ratio", domainGDP, "Derived GDP", estimatorGDPWorkerD,
    "Derivation worker ratio to GDP")

final <- data.frame(GDP = GDP, PoInst = PoInst)
final <- final[complete.cases(final), ]

estimatorGDPPoInst <- localPolynomsEstimation(final$PoInst, final$GDP, ordre, dimension,
    noyau, blocksize, 0.005, 1, 0.005)
plotHelper1D(final$PoInst, domainPoInst, "Politicial instability", final$GDP, domainGDP,
    "GDP", estimatorGDPPoInst, "Political instability to GDP")

estimatorGDPPoInstD <- derive1D(estimatorGDPPoInst, h)
plotHelperD1D(domainPoInst, "Political instability", domainGDP, "Derived GDP", estimatorGDPPoInstD,
    "Derivation political instability to GDP")

final <- data.frame(GDP = GDPRights, CivLibs = CivLibs)
final <- final[complete.cases(final), ]

estimatorGDPCivLibs <- localPolynomsEstimation(final$CivLibs, final$GDP, ordre, dimension,
    noyau, blocksize, 0.005, 1, 0.005)
plotHelper1D(final$CivLibs, domainCivLibs, "Civil liberties", final$GDP, domainGDPRights,
    "GDP", estimatorGDPCivLibs, "Civil liberties to GDP")
```

```
final <- data.frame(GDP = GDPRights, PoRights = PoRights)
final <- final[complete.cases(final), ]

estimatorGDPPoRights <- localPolynomsEstimation(final$PoRights, final$GDP, ordre,
    dimension, noyau, blocksize, 0.005, 1, 0.005)
plotHelper1D(final$PoRights, domainPoRights, "Political rights", final$GDP, domainGDPRights,
    "GDP", estimatorGDPPoRights, "Political rights to GDP")


dimension <- 2
ordre <- 1
blocksize <- 25

final <- data.frame(GDP = GDP, Import = Import, Export = Export)
final <- final[complete.cases(final), ]
plot(final$Export * (domainExport[2] - domainExport[1]) + domainExport[1], final$Import *
    (domainImport[2] - domainImport[1]) + domainImport[1], xlab = "Export ratio to GDP",
    ylab = "Import ratio to GDP")
title(main = "Export ratio and import ratio points")

exportImport <- rbind(final$Export, final$Import)
estimatorGDPExpImp <- localPolynomsEstimation(exportImport, final$GDP, ordre, dimension,
    gaussianNoyau(dimension), blocksize, 0.005, 1, 0.005)
plotHelper2D(domainExport, "\nExport ratio", domainImport, "\nImport ratio", estimatorGDPExpImp,
    domainGDP, "\n\n\nGDP", "Export ratio and Import ratio to GDP", xstart = 0,
    xend = 1, ystart = 0, yend = 1)

final <- data.frame(GDP = GDP, Export = Export, Life.expectation = Life.expectation)
final <- final[complete.cases(final), ]
plot(final$Export * (domainExport[2] - domainExport[1]) + domainExport[1], final$Life.expectation *
    (domainLifeExpectation[2] - domainLifeExpectation[1]) + domainLifeExpectation[1],
    xlab = "Export ratio", ylab = "Life expectation", main = "Export ratio and life expectation points")
exportLife <- rbind(final$Export, final$Life.expectation)
estimatorGDPExpLife <- localPolynomsEstimation(exportLife, final$GDP, ordre, dimension,
    gaussianNoyau(dimension), blocksize, 0.005, 1, 0.005)
plotHelper2D(domainExport, "\nExport ratio", domainLifeExpectation, "\nLife expectation",
    estimatorGDPExpLife, domainGDP, "\n\n\nGDP", "Export ratio and life expectation to GDP")


final <- data.frame(GDP = GDP, Export = Export, Worker = Worker)
final <- final[complete.cases(final), ]
plot(final$Export * (domainExport[2] - domainExport[1]) + domainExport[1], final$Worker *
    (domainWorker[2] - domainWorker[1]) + domainWorker[1], xlab = "Export ratio",
    ylab = "Worker ratio", main = "Export ratio and Worker ratio points")
exportWorker <- rbind(final$Export, final$Worker)
estimatorGDPExpWorker <- localPolynomsEstimation(exportWorker, final$GDP, ordre,
    dimension, gaussianNoyau(dimension), blocksize, 0.005, 1, 0.005)
plotHelper2D(domainExport, "\nExport ratio", domainWorker, "\nWorker ratio", estimatorGDPExpWorker,
    domainGDP, "\n\n\nGDP", "Export ratio and worker ratio to GDP")

final <- data.frame(GDP = GDPRights, PoRights = PoRights, CivLibs = CivLibs)
final <- final[complete.cases(final), ]
plot(final$PoRights * (domainPoRights[2] - domainPoRights[1]) + domainPoRights[1],
    final$CivLibs * (domainCivLibs[2] - domainCivLibs[1]) + domainCivLibs[1], xlab = "Political rights",
    ylab = "Civil liberties", main = "Political rights and civil liberties points")
poRightsCivLibs <- rbind(final$PoRights, final$CivLibs)
estimatorGDPPoRCivL <- localPolynomsEstimation(poRightsCivLibs, final$GDP, ordre,
    dimension, gaussianNoyau(dimension), blocksize, 0.005, 1, 0.005)
plotHelper2D(domainPoRights, "\nPolitical rights", domainCivLibs, "\nCivil liberties",
    estimatorGDPPoRCivL, domainGDPRights, "\n\n\nGDP", "Political rights and civil liberties to GDP")


# localPolynomsEstimation.R
#
# This file contains the function localPolynomsEstimation which realizes
# a local polynoms estimation.

source("crossValidation.R")

# function localPolynomsEstimation
#
# This function calculates via the method of cross validation a local polynoms
# estimator for the observations (xdatapoints, ydatapoints). For this purpose,
# it calculates for every h, starting at hstart, stepsize hstep and ending at hend,
# the estimation for the risk and selects the estimator with the minimal risk.
#
# @param xdatapoints multidimensional x-values --> "dimensions" x "number observations"
# @param ydatapoints 1 dimensional y-values --> 1 x "number observations"
# @param ordre Ordre of the local polynoms
# @param dimension Dimension of the data
# @param noyau Function pointer to a kernel
# @param blocksize Block size which is used for the cross validation
# @param hstart Start value for h for the incremental search procedure
# @param hend End value for h for the incremental search procedure
# @param hstep Step widht for the incremental search procedure of h
#
# @return Function pointer to the estimator using local polynoms. It takes as
# argument a vector of size 1 x "dimension" for which it calculates the estimated
# value
localPolynomsEstimation <- function(xdatapoints, ydatapoints, ordre, dimension, noyau,
    blocksize, hstart, hend, hstep) {

    minValue <- 2^109
    minParameter <- 0

    for (h in seq(hstart, hend, hstep)) {
        value <- crossValidation(xdatapoints, ydatapoints, localPolynomsEstimatorGenerator(ordre,
            h, noyau, dimension), dimension, blocksize)
        if (value < minValue) {
            minValue <- value
            minParameter <- h
        }
        cat("h:", h, " value:", value, "\n")
```

```
    }
    cat("min_h:", minParameter, "\n")
    localPolynomsEstimatorGenerator(ordre, minParameter, noyau, dimension)(xdatapoints,
        ydatapoints)
}


# leastSquaresEstimation.R
#
# This file contains a helper function for the least squares estimation
source("crossValidation.R")

leastSquaresEstimation <- function(xdatapoints, ydatapoints, dimension, dict, blocksize,
    mstart, mend) {

    minValue <- 2^109
    minParameter <- 0

    for (i in seq(mstart, mend)) {
        value <- crossValidation(xdatapoints, ydatapoints, leastSquaresEstimatorGenerator(dict,
            i), dimension, blocksize)
        if (value < minValue) {
            minValue <- value
            minParameter <- i
        }
        cat("m:", i, " value:", value, "\n")
    }
    cat("min_m:", minParameter, "\n")
    leastSquaresEstimatorGenerator(dict, minParameter)(xdatapoints, ydatapoints)
}


# crossValidation.R
#
# function crossValidation
#
# This function performs a cross validation to estimate the risk for a given
# estimator generator.
#
# @param xdatapoints x-values of the observations
# @param ydatapoints y-values of the observations
# @param estimatorGenerator Generator function for the estimator
# @param dimension dimension of the x-values
# @param blocksize size of the blocks for the cross-validation
#
# @return estimated risk value
crossValidation <- function(xdatapoints, ydatapoints, estimatorGenerator, dimension,
    blocksize = 1) {
    risk <- 0
    numdatapoints <- length(xdatapoints)/dimension
    dim(xdatapoints) = c(dimension, numdatapoints)

    # mix the data
    xdatapoints <- xdatapoints[, sample.int(numdatapoints)]

    dim(xdatapoints) = c(dimension, numdatapoints)

    # for every block of test data do
    for (index in seq(1, numdatapoints, by = blocksize)) {
        xtestpoints <- xdatapoints[, index:min(numdatapoints, index + blocksize -
            1)]
        ytestpoints <- ydatapoints[index:min(numdatapoints, index + blocksize - 1)]

        dim(xtestpoints) = c(dimension, min(numdatapoints - index + 1, blocksize))
        xlearningpoints <- xdatapoints[, -(index:min(numdatapoints, index + blocksize -
            1))]
        ylearningpoints <- ydatapoints[-(index:min(numdatapoints, index + blocksize -
            1))]

        dim(xlearningpoints) = c(dimension, numdatapoints - min(numdatapoints - index +
            1, blocksize))
        # estimator for current learning data set
        estimator <- estimatorGenerator(xlearningpoints, ylearningpoints)

        numblockdatapoints <- min(blocksize, numdatapoints - index + 1)
        # the risk for the corresponding test data set
        temprisk <- crossprod((ytestpoints - estimator(xtestpoints)), (ytestpoints -
            estimator(xtestpoints)))
        risk <- risk + temprisk/numblockdatapoints
    }

    risk
}
```