

# Package ‘ElemStatLearn’

January 2, 2012

**Date** 2009-10-10

**Title** Data sets, functions and examples from the book: “The Elements of Statistical Learning, Data Mining, Inference, and Prediction” by Trevor Hastie, Robert Tibshirani and Jerome Friedman.

**Version** 0.1-7.1

**Author** Material from the book’s webpage, R port and packaging by Kjetil Halvorsen

**Description** Useful when reading the book above mentioned, in the documentation referred to as ‘the book’.

**Depends** R ( $\geq 2.10$ ), stats

**Suggests** gam, splines, MASS, class, mclust02, leaps, mda, lasso2,lars, boot

**LazyData** yes

**LazyDataCompression** xz

**Maintainer** Kjetil Halvorsen <kjetil1001@gmail.com>

**URL**

**License** GPL ( $\geq 2$ )

**Repository** CRAN

**Date/Publication** 2011-02-11 10:05:23

## R topics documented:

bone . . . . .	2
countries . . . . .	3
galaxy . . . . .	4
marketing . . . . .	5
mixture.example . . . . .	6
nci . . . . .	9
orange10.test . . . . .	10

orange10.train . . . . .	11
orange4.test . . . . .	11
orange4.train . . . . .	12
ozone . . . . .	13
phoneme . . . . .	13
prostate . . . . .	21
SAheart . . . . .	24
simple.ridge . . . . .	25
spam . . . . .	26
vowel.test . . . . .	29
vowel.train . . . . .	30
waveform . . . . .	35
waveform.test . . . . .	36
waveform.train . . . . .	37
zip.test . . . . .	38
zip.train . . . . .	38
zip2image . . . . .	39

<b>Index</b>	<b>41</b>
--------------	-----------

---

bone	<i>Bone Mineral Density Data</i>
------	----------------------------------

---

## Description

Measurements in the bone mineral density of 261 north american adolescents, as function of age. Each value is the difference in spnbmd taken on two consecutive visits, divided by the average. The age is the average age over the two visits.

## Usage

```
data(bone)
```

## Format

A data frame with 485 observations on the following 4 variables.

**idnum** identifies the child, and hence the repeat measurements

**age** average of age at two visits

**gender** a factor with levels female male

**spnbmd** Relative Spinal bone mineral density measurement

## Source

Bachrach LK, Hastie T, Wang M-C, Narasimhan B, Marcus R. Bone Mineral Acquisition in Healthy Asian, Hispanic, Black and Caucasian Youth. A Longitudinal Study. *J Clin Endocrinol Metab* (1999) 84, 4702-12.

**Examples**

```
summary(bone)
# We ignore the repeat measurements:
# smooth.spline is in standard package stats.
names(bone)
# Plot page 128 in the book:
plot(spnbmd ~ age, data=bone, col =
     ifelse(gender=="male", "blue", "red2"),
     xlab="Age", ylab="Relative Change in Spinal BMD")
bone.spline.male <- with(subset(bone,gender=="male"),
     smooth.spline(age, spnbmd,df=12))
bone.spline.female <- with(subset(bone, gender=="female"),
     smooth.spline(age, spnbmd, df=12))
lines(bone.spline.male, col="blue")
lines(bone.spline.female, col="red2")
legend(20,0.20, legend=c("male", "Female"), col=c("blue", "red2"),
     lwd=2)
```

---

countries

*Country Dissimilarities*

---

**Description**

Political science students were asked to give pairwise dissimilarities for 12 countries.

**Usage**

```
data(countries)
```

**Format**

The format is: num [1:12, 1:12] 0 5.58 7 7.08 4.83 2.17 6.42 3.42 2.5 6.08 ... - attr(\*, "dim-names")=List of 2 ..\$ : chr [1:12] "BEL" "BRA" "CHI" "CUB" ... ..\$ : chr [1:12] "BEL" "BRA" "CHI" "CUB" ...

**Details**

These are average dissimilarities over students.

**Source**

Kaufman, L. and Rousseeuw, P. (1990) Finding Groups in data: An Introduction to Cluster Analysis, Wiley, New York.

**Examples**

```

str(countries)
colnames(countries)
rownames(countries)
if(require(MASS)){
# We use multidimensional scaling:
  if(interactive())par(ask=TRUE)
  countries.cmdscale <- cmdscale(countries, k=2, eig=TRUE)
  eqscplot(countries.cmdscale$points)
  countries.sam <- sammon(countries)
  eqscplot(countries.sam$points)
}

```

---

galaxy

*Galaxy Data*


---

**Description**

Radial Velocity of Galaxy NGC7531

**Usage**

```
data(galaxy)
```

**Format**

A data frame with 323 observations on the following 5 variables.

**east.west** the east-west coordinate. The origin, (0,0), is near the center of the galaxy, east is negative, west is positive.

**north.south** the north-south coordinate. The origin, (0,0), is near the center of the galaxy, south is negative, north is positive.

**angle** degrees of counter-clockwise rotation from the horizontal of the slot within which the observation lies.

**radial.position** signed distance from origin; negative if east-west coordinate is negative.

**velocity** radial velocity measured in km/sec.

**Details**

The galaxy data frame records the radial velocity of a spiral galaxy measured at 323 points in the area of sky which it covers. All the measurements lie within seven slots crossing at the origin. The positions of the measurements given by four variables (columns).

**Source**

Buta, R. (1987) The Structure and Dynamics of Ringed Galaxies, III: Surface Photometry and Kinematics of the Ringed Nonbarred Spiral NGC7531. The Astrophysical J. Supplement Ser. Vol. 64, pp. 1–37.

John M. Chambers and Trevor J. Hastie, (eds.) Statistical Models in S, Wadsworth and Brooks, Pacific Grove, CA 1992, pg. 352.

**Examples**

```
str(galaxy)
summary(galaxy)
```

---

marketing	<i>Market Basket Analysis</i>
-----------	-------------------------------

---

**Description**

The dataset is an extract from this survey. It consists of 14 demographic attributes. The dataset is a good mixture of categorical and continuous variables with a lot of missing data. This is characteristic for data mining applications.

**Usage**

```
data(marketing)
```

**Format**

A data frame with 8993 observations on the following 14 variables.

**Income** ANNUAL INCOME OF HOUSEHOLD (PERSONAL INCOME IF SINGLE) 1. Less than \$10,000 2. \$10,000 to \$14,999 3. \$15,000 to \$19,999 4. \$20,000 to \$24,999 5. \$25,000 to \$29,999 6. \$30,000 to \$39,999 7. \$40,000 to \$49,999 8. \$50,000 to \$74,999 9. \$75,000 or more

**Sex** 1. Male 2. Female

**Marital** 1. Married 2. Living together, not married 3. Divorced or separated 4. Widowed 5. Single, never married

**Age** 1. 14 thru 17 2. 18 thru 24 3. 25 thru 34 4. 35 thru 44 5. 45 thru 54 6. 55 thru 64 7. 65 and Over

**Edu** 1. Grade 8 or less 2. Grades 9 to 11 3. Graduated high school 4. 1 to 3 years of college 5. College graduate 6. Grad Study

**Occupation** 1. Professional/Managerial 2. Sales Worker 3. Factory Worker/Laborer/Driver 4. Clerical/Service Worker 5. Homemaker 6. Student, HS or College 7. Military 8. Retired 9. Unemployed

**Lived** HOW LONG HAVE YOU LIVED IN THE SAN FRAN./OAKLAND/SAN JOSE AREA? 1. Less than one year 2. One to three years 3. Four to six years 4. Seven to ten years 5. More than ten years

**Dual\\_Income** DUAL INCOMES (IF MARRIED) 1. Not Married 2. Yes 3. No

**Household** PERSONS IN YOUR HOUSEHOLD 1. One 2. Two 3. Three 4. Four 5. Five 6. Six 7. Seven 8. Eight 9. Nine or more

**Householdu18** PERSONS IN HOUSEHOLD UNDER 18 0. None 1. One 2. Two 3. Three 4. Four 5. Five 6. Six 7. Seven 8. Eight 9. Nine or more

**Status** HOUSEHOLDER STATUS 1. Own 2. Rent 3. Live with Parents/Family

**Home\\_Type** 1. House 2. Condominium 3. Apartment 4. Mobile Home 5. Other

**Ethnic** 1. American Indian 2. Asian 3. Black 4. East Indian 5. Hispanic 6. Pacific Islander 7. White 8. Other

**Language** WHAT LANGUAGE IS SPOKEN MOST OFTEN IN YOUR HOME? 1. English 2. Spanish 3. Other

### Details

The goal is to predict the Annual Income of Household from the other 13 demographics attributes.

Number of instances: 8993.

These are obtained from the original dataset with 9409 instances, by removing those observations with the response (Annual Income) missing.

### Source

Source: Impact Resources, Inc., Columbus, OH (1987). A total of N=9409 questionnaires containing 502 questions were filled out by shopping mall customers in the San Francisco Bay area.

### Examples

```
str(marketing)
summary(marketing)
```

---

mixture.example

*Mixture Example*

---

### Description

This is a simulated mixture example with 200 instances and two classes. 100 members in each class.

### Usage

```
data(mixture.example)
```

**Format**

The format is: List of 8 \ \$ x : 200 x 2 matrix of training predictors \ \$ y : 200 x 2 matrix of class labels, 0==green, 1==red \ \$ xnew : matrix [1:6831, 1:2] -2.6 -2.5 -2.4 -2.3 -2.2 ... ..- attr(\*, "class")= chr "matrix" ..- attr(\*, "dimnames")=List of 2 .. ..\$ : chr [1:6831] "1" "2" "3" "4" ... .. \ \$ : chr [1:2] "x1" "x2" : matrix 6831 x 2 of lattice points in predictor space \ \$ prob : atomic [1:6831] 3.55e-05 3.05e-05 2.63e-05 2.27e-05 1.96e-05 ... ..- attr(\*, ".Names")= chr [1:6831] "1" "2" "3" "4" ... vector of 6831 probabilities (of class RED) at each lattice point \ \$ marginal: atomic [1:6831] 6.65e-15 2.31e-14 7.62e-14 2.39e-13 7.15e-13 ... ..- attr(\*, ".Names")= chr [1:6831] "1" "2" "3" "4" ... : marginal probability at each lattice point \ \$ px1 : 69 lattice coordinates for x.1 \ \$ px2 : 99 lattice values for x.2 (69\*99=6831) \ \$ means : num [1:20, 1:2] : 20 x 2 matrix of the mixture centers, first ten for one class, next ten for the other

**Examples**

```
str(mixture.example)
if(interactive())par(ask=TRUE)
x <- mixture.example$x
g <- mixture.example$y
x.mod <- lm( g ~ x)
# Figure 2.1:
plot(x, col=ifelse(g==1,"red", "green"), xlab="x1", ylab="x2")
coef(x.mod)
abline( (0.5-coef(x.mod)[1])/coef(x.mod)[3], -coef(x.mod)[2]/coef(x.mod)[3])
ghat <- ifelse( fitted(x.mod)>0.5, 1, 0)
length(ghat)

sum(ghat == g)

1 - sum(ghat==g)/length(g)
#[1] 0.27
# Training misclassification rate
xnew <- mixture.example$xnew
dim(xnew)
colnames(xnew)

library(class)

mod15 <- knn(x, xnew, g, k=15, prob=TRUE)
summary(mod15)
#Figure 2.2:
plot(x, col=ifelse(g==1,"red", "green"),xlab="x1", ylab="x2")

str(mod15)
prob <- attr(mod15, "prob")
prob <- ifelse( mod15=="1", prob, 1-prob) # prob is voting fraction for winning class!
# Now it is voting fraction for red==1

px1 <- mixture.example$px1
px2 <- mixture.example$px2

prob15 <- matrix(prob, length(px1), length(px2))
```

```

contour(px1, px2, prob15, levels=0.5, labels="", xlab="x1", ylab="x2", main=
  "15-nearest neighbour")
# adding the points to the plot:
points(x, col=ifelse(g==1, "red", "green"))

ghat15 <- ifelse(knn(x,x,k=15, cl=g)=="1", 1, 0)
sum(ghat15==g)
# [1] 169
1 - sum(ghat15==g)/length(g)
# [1] 0.155
# Misclassification rate for knn(, k=15)

# Then we want the plot for knn with k=1: (Figure 2.3)

mod1 <- knn(x, xnew, k=1, cl=g, prob=TRUE)
prob <- attr(mod1, "prob")
prob <- ifelse( mod1=="1", prob, 1-prob) # prob now is voting
                                         # fraction for "red"
prob1 <- matrix(prob, length(px1), length(px2) )
contour(px1, px2, prob1, level=0.5, labels="", xlab="x1", ylab="x2", main=
  "1-nearest neighbour")
# Adding the points to the plot:
points(x, col=ifelse(g==1, "red", "green"))

# Reproducing figure 2.4, page 17 of the book:
# The data do not contain a test sample, so we make one,
# using the description of the oracle page 17 of the book: The centers
# is in the means component of mixture.example, with green(0) first,
# so red(1). For a test sample of size 10000 we simulate
# 5000 observations of each class.

library(MASS)
set.seed(123)
centers <- c(sample(1:10, 5000, replace=TRUE),
             sample(11:20, 5000, replace=TRUE))
means <- mixture.example$means
means <- means[centers, ]
mix.test <- mvrnorm(10000, c(0,0), 0.2*diag(2))
mix.test <- mix.test + means
cltest <- c(rep(0, 5000), rep(1, 5000))

ks <- c(1,3,5,7,9,11,15,17,23,25,35,45,55,83,101,151 )
# nearest neighbours to try
nks <- length(ks)
misclass.train <- numeric(length=nks)
misclass.test <- numeric(length=nks)
names(misclass.train) <- names(misclass.test) <- ks
for (i in seq(along=ks)) {
  mod.train <- knn(x,x,k=ks[i],cl=g)
  mod.test <- knn(x, mix.test,k= ks[i],cl= g)
  misclass.train[i] <- 1 - sum(mod.train==factor(g))/200
  misclass.test[i] <- 1 - sum(mod.test==factor(cltest))/10000
}

```

```

    }
    print(cbind(misclass.train, misclass.test))

# Using package mclust02 # Note that this package is no longer on CRAN,
#                       but must be searched in the archives.

if(require(mclust02)){
  x <- mixture.example$x
  g <- mixture.example$y
  xnew <- mixture.example$xnew
  px1 <- mixture.example$px1
  px2 <- mixture.example$px2

  mix.mclust <- mclustDA(x, g, xnew, G=1:6, verbose=TRUE)
  mix.mclust
} # end require (mclust02)

# Figure 2.4
plot(misclass.train,xlab="Number of NN",ylab="Test error",type="n",xaxt="n")
axis(1, 1:length(ks), as.character(ks))
lines(misclass.test,type="b",col='blue',pch=20)
lines(misclass.train,type="b",col='red',pch=20)
legend("bottomright",lty=1,col=c("red","blue"),legend = c("train ", "test "))

#Figure 2.5
prob<-mixture.example$prob
prob.bayes <- matrix(prob, length(px1), length(px2))
contour(px1, px2, prob.bayes, levels=0.5, labels="", xlab="x1", ylab="x2", main="Bayes decision boundary")
points(x, col=ifelse(g==1, "red", "green"))

```

---

nci

*NCI microarray data (chap 14)*


---

## Description

Human Tumour Microarray Data

## Usage

```
data(nci)
```

## Format

The format is: num [1:6830, 1:64] 0.300 1.180 0.550 1.140 -0.265 ... - attr(\*, "dimnames")=List of 2 ..\$ : NULL ..\$ : chr [1:64] "CNS" "CNS" "CNS" "RENAL" ...

## Source

<http://genome-www.stanford.edu/nci60/>

**Examples**

```
str(nci)
summary(nci)
```

---

orange10.test

*Simulated Orange Data*


---

**Description**

These simulation data are described on the revised page 384-5 of the book. See the errata file for a revised version if your book is not the fourth or later printing.

There are 50 x four datasets - training and test data for the four dimensional problem, and training and test data for the 10 dimensional problem.

**Usage**

```
data(orange10.test)
```

**Format**

The format is: List of 50 `'data.frame'`: 1000 obs. of 12 variables: `..$ class: int [1:1000] -1 -1 -1 -1 -1 -1 -1 -1 ... ..$ F1 : num [1:1000] -0.780 2.097 0.798 -0.896 -0.365 ... ..$ F2 : num [1:1000] 2.5767 0.4146 0.0381 -2.0925 -2.2895 ... ..$ F3 : num [1:1000] 1.18 -2.45 -2.03 -1.36 -2.63 ... ..$ F4 : num [1:1000] -0.997 0.710 -2.223 2.478 -0.666 ... ..$ F5 : num [1:1000] -0.308 0.418 0.407 -0.198 1.347 ... ..$ F6 : num [1:1000] 0.126 -0.718 -1.173 2.390 0.122 ... ..$ F7 : num [1:1000] 0.3233 0.0846 0.3814 -0.5928 -0.0555 ... ..$ F8 : num [1:1000] -0.668 1.836 2.006 -0.782 -0.164 ... ..$ F9 : num [1:1000] -1.7559 -0.0389 1.7472 -2.1746 0.5236 ... ..$ F10 : num [1:1000] -0.926 0.196 -0.960 -0.820 -1.207 ... ..$ f : Factor w/ 50 levels "1","2","3","4",,..: 1 1 1 1 1 1 1 1 1 1 ... ..` and then 49 similar ones.

**Source**

Data simulated for the book.

**Examples**

```
str(orange10.test)
```

---

orange10.train	<i>Simulated Orange Data</i>
----------------	------------------------------

---

**Description**

These simulation data are described on the revised page 384-5 of the book. See the errata file for a revised version if your book is not the fourth or later printing.

There are 50 x four datasets - training and test data for the four dimensional problem, and training and test data for the 10 dimensional problem.

**Usage**

```
data(orange10.train)
```

**Format**

The format is: List of 50 \ \$ 1 : 'data.frame': 100 obs. of 12 variables: .. \$ class: int [1:100] -1 -1 -1 -1 -1 -1 -1 -1 ... .. \$ F1 : num [1:100] -3.063 0.488 -2.791 2.010 -0.961 ... .. \$ F2 : num [1:100] -1.091 -1.025 0.180 0.875 -2.388 ... .. \$ F3 : num [1:100] 0.0466 -1.5788 -1.5160 1.8155 -1.7695 ... .. \$ F4 : num [1:100] -1.794 -2.314 -1.013 -2.374 -0.857 ... .. \$ F5 : num [1:100] 1.2722 2.0690 -0.0983 -2.2647 -0.7894 ... .. \$ F6 : num [1:100] 1.393 -0.398 -0.930 -1.687 -1.121 ... .. \$ F7 : num [1:100] -2.433 0.964 -0.387 0.851 -1.163 ... .. \$ F8 : num [1:100] 0.610 -0.621 -0.727 -0.319 1.914 ... .. \$ F9 : num [1:100] 0.904 -1.129 -0.377 -0.665 -0.691 ... .. \$ F10 : num [1:100] 1.591 0.533 0.837 1.285 -1.056 ... .. \$ f : Factor w/ 50 levels "1","2","3","4",...: 1 1 1 1 1 1 1 1 1 1 ... .. and then 49 similar ones.

**Source**

Data simulated for the book.

**Examples**

```
str(orange10.train)
```

---

orange4.test	<i>Simulated Orange Data</i>
--------------	------------------------------

---

**Description**

These simulation data are described on the revised page 384-5 of the book. See the errata file for a revised version if your book is not the fourth or later printing.

There are 50 x four datasets - training and test data for the four dimensional problem, and training and test data for the 10 dimensional problem.

**Usage**

```
data(orange4.test)
```

**Format**

The format is: List of 50 \ \$ 1 :'data.frame': 1000 obs. of 6 variables: ..\$ class: int [1:1000] -1 -1 -1 -1 -1 -1 -1 -1 -1 ... ..\$ F1 : num [1:1000] 2.352 -0.295 1.000 -0.133 -0.774 ... ..\$ F2 : num [1:1000] -1.56 -2.08 1.59 -1.39 0.25 ... ..\$ F3 : num [1:1000] -1.058 0.355 1.779 2.281 1.486 ... ..\$ F4 : num [1:1000] -0.0382 2.2957 -1.8081 1.5588 -3.0523 ... ..\$ f : Factor w/ 50 levels "1","2","3","4",...: 1 1 1 1 1 1 1 1 1 1 ... .. and then 49 similar ones.

**Source**

Data simulated for the book.

**Examples**

```
str(orange4.test)
```

---

```
orange4.train
```

```
Simulated Orange Data
```

---

**Description**

These simulation data are described on the revised page 384-5 of the book. See the errata file for a revised version if your book is not the fourth or later printing.

There are 50 x four datasets - training and test data for the four dimensional problem, and training and test data for the 10 dimensional problem.

**Usage**

```
data(orange4.train)
```

**Format**

The format is: List of 50 \ \$ 1 :'data.frame': 100 obs. of 6 variables: ..\$ class: int [1:100] -1 -1 -1 -1 -1 -1 -1 -1 -1 ... ..\$ F1 : num [1:100] -0.252 -2.482 -2.497 2.220 0.727 ... ..\$ F2 : num [1:100] 2.73 1.48 -2.13 -1.64 -2.62 ... ..\$ F3 : num [1:100] -0.864 0.411 0.136 -1.809 1.344 ... ..\$ F4 : num [1:100] 1.535 1.335 0.482 1.682 0.834 ... ..\$ f : Factor w/ 50 levels "1","2","3","4",...: 1 1 1 1 1 1 1 1 1 1 ... .. and then 49 similar ones.

**Source**

Data simulated for the book.

**Examples**

```
str(orange4.train)
```

---

ozone

*Ozone Data*

---

### Description

Ozone data from New York.

### Usage

```
data(ozone)
```

### Format

A data frame with 111 observations on the following 4 variables.

**ozone** a numeric vector

**radiation** a numeric vector

**temperature** a numeric vector

**wind** a numeric vector

### Details

Data frame with components: ozone, radiation, temperature, and wind. Measurements of daily ozone concentration (ppb), wind speed (mph), daily maximum temperature (degrees F), and solar radiation (langleys) on 111 days from May to September 1973 in New York. This data frame is similar to `air` in `S-PLUS` (or `library(data)` in `S`), but has a different definition for ozone (`air` contains cube-roots of ozone).

### Examples

```
str(ozone)
plot(ozone)
```

---

phoneme

*Data From a Acoustic-Phonetic Continuous Speech Corpus*

---

### Description

See Details.

### Usage

```
data(phoneme)
```

**Format**

A data frame with 4509 observations on the following 258 variables.

- x.1** a numeric vector
- x.2** a numeric vector
- x.3** a numeric vector
- x.4** a numeric vector
- x.5** a numeric vector
- x.6** a numeric vector
- x.7** a numeric vector
- x.8** a numeric vector
- x.9** a numeric vector
- x.10** a numeric vector
- x.11** a numeric vector
- x.12** a numeric vector
- x.13** a numeric vector
- x.14** a numeric vector
- x.15** a numeric vector
- x.16** a numeric vector
- x.17** a numeric vector
- x.18** a numeric vector
- x.19** a numeric vector
- x.20** a numeric vector
- x.21** a numeric vector
- x.22** a numeric vector
- x.23** a numeric vector
- x.24** a numeric vector
- x.25** a numeric vector
- x.26** a numeric vector
- x.27** a numeric vector
- x.28** a numeric vector
- x.29** a numeric vector
- x.30** a numeric vector
- x.31** a numeric vector
- x.32** a numeric vector
- x.33** a numeric vector
- x.34** a numeric vector
- x.35** a numeric vector

- x.36** a numeric vector
- x.37** a numeric vector
- x.38** a numeric vector
- x.39** a numeric vector
- x.40** a numeric vector
- x.41** a numeric vector
- x.42** a numeric vector
- x.43** a numeric vector
- x.44** a numeric vector
- x.45** a numeric vector
- x.46** a numeric vector
- x.47** a numeric vector
- x.48** a numeric vector
- x.49** a numeric vector
- x.50** a numeric vector
- x.51** a numeric vector
- x.52** a numeric vector
- x.53** a numeric vector
- x.54** a numeric vector
- x.55** a numeric vector
- x.56** a numeric vector
- x.57** a numeric vector
- x.58** a numeric vector
- x.59** a numeric vector
- x.60** a numeric vector
- x.61** a numeric vector
- x.62** a numeric vector
- x.63** a numeric vector
- x.64** a numeric vector
- x.65** a numeric vector
- x.66** a numeric vector
- x.67** a numeric vector
- x.68** a numeric vector
- x.69** a numeric vector
- x.70** a numeric vector
- x.71** a numeric vector
- x.72** a numeric vector

**x.73** a numeric vector  
**x.74** a numeric vector  
**x.75** a numeric vector  
**x.76** a numeric vector  
**x.77** a numeric vector  
**x.78** a numeric vector  
**x.79** a numeric vector  
**x.80** a numeric vector  
**x.81** a numeric vector  
**x.82** a numeric vector  
**x.83** a numeric vector  
**x.84** a numeric vector  
**x.85** a numeric vector  
**x.86** a numeric vector  
**x.87** a numeric vector  
**x.88** a numeric vector  
**x.89** a numeric vector  
**x.90** a numeric vector  
**x.91** a numeric vector  
**x.92** a numeric vector  
**x.93** a numeric vector  
**x.94** a numeric vector  
**x.95** a numeric vector  
**x.96** a numeric vector  
**x.97** a numeric vector  
**x.98** a numeric vector  
**x.99** a numeric vector  
**x.100** a numeric vector  
**x.101** a numeric vector  
**x.102** a numeric vector  
**x.103** a numeric vector  
**x.104** a numeric vector  
**x.105** a numeric vector  
**x.106** a numeric vector  
**x.107** a numeric vector  
**x.108** a numeric vector  
**x.109** a numeric vector

- x.110** a numeric vector
- x.111** a numeric vector
- x.112** a numeric vector
- x.113** a numeric vector
- x.114** a numeric vector
- x.115** a numeric vector
- x.116** a numeric vector
- x.117** a numeric vector
- x.118** a numeric vector
- x.119** a numeric vector
- x.120** a numeric vector
- x.121** a numeric vector
- x.122** a numeric vector
- x.123** a numeric vector
- x.124** a numeric vector
- x.125** a numeric vector
- x.126** a numeric vector
- x.127** a numeric vector
- x.128** a numeric vector
- x.129** a numeric vector
- x.130** a numeric vector
- x.131** a numeric vector
- x.132** a numeric vector
- x.133** a numeric vector
- x.134** a numeric vector
- x.135** a numeric vector
- x.136** a numeric vector
- x.137** a numeric vector
- x.138** a numeric vector
- x.139** a numeric vector
- x.140** a numeric vector
- x.141** a numeric vector
- x.142** a numeric vector
- x.143** a numeric vector
- x.144** a numeric vector
- x.145** a numeric vector
- x.146** a numeric vector

**x.147** a numeric vector  
**x.148** a numeric vector  
**x.149** a numeric vector  
**x.150** a numeric vector  
**x.151** a numeric vector  
**x.152** a numeric vector  
**x.153** a numeric vector  
**x.154** a numeric vector  
**x.155** a numeric vector  
**x.156** a numeric vector  
**x.157** a numeric vector  
**x.158** a numeric vector  
**x.159** a numeric vector  
**x.160** a numeric vector  
**x.161** a numeric vector  
**x.162** a numeric vector  
**x.163** a numeric vector  
**x.164** a numeric vector  
**x.165** a numeric vector  
**x.166** a numeric vector  
**x.167** a numeric vector  
**x.168** a numeric vector  
**x.169** a numeric vector  
**x.170** a numeric vector  
**x.171** a numeric vector  
**x.172** a numeric vector  
**x.173** a numeric vector  
**x.174** a numeric vector  
**x.175** a numeric vector  
**x.176** a numeric vector  
**x.177** a numeric vector  
**x.178** a numeric vector  
**x.179** a numeric vector  
**x.180** a numeric vector  
**x.181** a numeric vector  
**x.182** a numeric vector  
**x.183** a numeric vector

- x.184** a numeric vector
- x.185** a numeric vector
- x.186** a numeric vector
- x.187** a numeric vector
- x.188** a numeric vector
- x.189** a numeric vector
- x.190** a numeric vector
- x.191** a numeric vector
- x.192** a numeric vector
- x.193** a numeric vector
- x.194** a numeric vector
- x.195** a numeric vector
- x.196** a numeric vector
- x.197** a numeric vector
- x.198** a numeric vector
- x.199** a numeric vector
- x.200** a numeric vector
- x.201** a numeric vector
- x.202** a numeric vector
- x.203** a numeric vector
- x.204** a numeric vector
- x.205** a numeric vector
- x.206** a numeric vector
- x.207** a numeric vector
- x.208** a numeric vector
- x.209** a numeric vector
- x.210** a numeric vector
- x.211** a numeric vector
- x.212** a numeric vector
- x.213** a numeric vector
- x.214** a numeric vector
- x.215** a numeric vector
- x.216** a numeric vector
- x.217** a numeric vector
- x.218** a numeric vector
- x.219** a numeric vector
- x.220** a numeric vector

**x.221** a numeric vector  
**x.222** a numeric vector  
**x.223** a numeric vector  
**x.224** a numeric vector  
**x.225** a numeric vector  
**x.226** a numeric vector  
**x.227** a numeric vector  
**x.228** a numeric vector  
**x.229** a numeric vector  
**x.230** a numeric vector  
**x.231** a numeric vector  
**x.232** a numeric vector  
**x.233** a numeric vector  
**x.234** a numeric vector  
**x.235** a numeric vector  
**x.236** a numeric vector  
**x.237** a numeric vector  
**x.238** a numeric vector  
**x.239** a numeric vector  
**x.240** a numeric vector  
**x.241** a numeric vector  
**x.242** a numeric vector  
**x.243** a numeric vector  
**x.244** a numeric vector  
**x.245** a numeric vector  
**x.246** a numeric vector  
**x.247** a numeric vector  
**x.248** a numeric vector  
**x.249** a numeric vector  
**x.250** a numeric vector  
**x.251** a numeric vector  
**x.252** a numeric vector  
**x.253** a numeric vector  
**x.254** a numeric vector  
**x.255** a numeric vector  
**x.256** a numeric vector  
**g** a factor with levels aa ao dcl iy sh  
**speaker** a factor with 437 levels

## Details

These data arose from a collaboration between Andreas Buja, Werner Stuetzle and Martin Maechler, and we used as an illustration in the paper on Penalized Discriminant Analysis by Hastie, Buja and Tibshirani (1995), referenced in the text.

The data were extracted from the TIMIT database (TIMIT Acoustic-Phonetic Continuous Speech Corpus, NTIS, US Dept of Commerce) which is a widely used resource for research in speech recognition. A dataset was formed by selecting five phonemes for classification based on digitized speech from this database. The phonemes are transcribed as follows: "sh" as in "she", "dcl" as in "dark", "iy" as the vowel in "she", "aa" as the vowel in "dark", and "ao" as the first vowel in "water". From continuous speech of 50 male speakers, 4509 speech frames of 32 msec duration were selected, approximately 2 examples of each phoneme from each speaker. Each speech frame is represented by 512 samples at a 16kHz sampling rate, and each frame represents one of the above five phonemes. The breakdown of the 4509 speech frames into phoneme frequencies is as follows:

```
aa ao dcl iy sh 695 1022 757 1163 872
```

From each speech frame, we computed a log-periodogram, which is one of several widely used methods for casting speech data in a form suitable for speech recognition. Thus the data used in what follows consist of 4509 log-periodograms of length 256, with known class (phoneme) memberships.

The data contain 256 columns labelled "x.1" - "x.256", a response column labelled "g", and a column labelled "speaker" identifying the different speakers.

## Examples

```
head(str(phoneme))
```

---

```
prostate
```

```
Prostate Cancer Data
```

---

## Description

Data to examine the correlation between the level of prostate-specific antigen and a number of clinical measures in men who were about to receive a radical prostatectomy.

## Usage

```
data(prostate)
```

## Format

A data frame with 97 observations on the following 10 variables.

**lcavol** log cancer volume

**lweight** log prostate weight

**age** in years

**lbph** log of the amount of benign prostatic hyperplasia

**svi** seminal vesicle invasion

**lcp** log of capsular penetration  
**gleason** a numeric vector  
**pgg45** percent of Gleason score 4 or 5  
**lpsa** response  
**train** a logical vector

### Details

The last column indicates which 67 observations were used as the "training set" and which 30 as the test set, as described on page 48 in the book.

### Source

Stamey, T., Kabalin, J., McNeal, J., Johnstone, I., Freiha, F., Redwine, E. and Yang, N (1989) Prostate specific antigen in the diagnosis and treatment of adenocarcinoma of the prostate II. Radical prostatectomy treated patients, *Journall of Urology* 16: 1076–1083.

### Examples

```
if(interactive())par(ask=TRUE)
str( prostate )
cor( prostate[,1:8] )
pairs( prostate[,1:9], col="violet" )
train <- subset( prostate, train==TRUE )[,1:9]
test  <- subset( prostate, train=FALSE )[,1:9]
#
if( require(leaps)) {
# The book (page 56) uses only train subset, so we the same:
prostate.leaps <- regsubsets( lpsa ~ . , data=train, nbest=70, #all!
                             really.big=TRUE )
prostate.leaps.sum <- summary( prostate.leaps )
prostate.models <- prostate.leaps.sum$which
prostate.models.size <- as.numeric(attr(prostate.models, "dimnames")[[1]])
hist( prostate.models.size )
prostate.models.rss <- prostate.leaps.sum$rss
prostate.models.best.rss <-
  tapply( prostate.models.rss, prostate.models.size, min )
prostate.models.best.rss
# Let us add results for the only intercept model
prostate.dummy <- lm( lpsa ~ 1, data=train )
prostate.models.best.rss <- c(
  sum(resid(prostate.dummy)^2),
  prostate.models.best.rss)
# Making a plot:
plot( 0:8, prostate.models.best.rss, ylim=c(0, 100),
      type="b", xlab="subset size", ylab="Residual Sum Square",
      col="red2" )
points( prostate.models.size, prostate.models.rss, pch=17, col="brown",cex=0.7 )
}
# For a better plot, should remove the best for each size from last call!
# Now with ridge regression:
```

```

# Ridge regression in R is multiply implemented, at least:
# MASS: lm.ridge
# mda : gen.ridge
#( survival: ridge)
# Design: pentrace
# mgcv: pcls (very general)
# simple.ridge (in this package)
#
library(mda)
#
prostate.ridge.list <- lapply(list(lambda=seq(0,8,by=0.4)), function(lambda)
  gen.ridge(train[,1:8], y=train[,9,drop=FALSE], lambda=lambda))
# Problems with this usage.
# simpler usage:
#
prostate.ridge <- gen.ridge(train[,1:8], y=train[,9,drop=FALSE], lambda=1)
#
# Since there is some problems with the mda functions, we use our own:
#
prostate.ridge <- simple.ridge( train[,1:8], train[,9], df=1:8 )
#
# coefficient traces:
#
matplot( prostate.ridge$df, t(prostate.ridge$beta), type="b",
  col="blue", pch=17, ylab="coefficients" )
# Calculations for the lasso:
#
if(require(lasso2)) {
prostate.lasso <- l1ce( lpsa ~ ., data=train, trace=TRUE, sweep.out=~1,
  bound=seq(0,1,by=0.1) )
prostate.lasso.coef <- sapply(prostate.lasso, function(x) x$coef)
colnames(prostate.lasso.coef) <- seq( 0,1,by=0.1 )
matplot( seq(0,1,by=0.1), t(prostate.lasso.coef[-1,]), type="b",
  xlab="shrinkage factor", ylab="coefficients",
  xlim=c(0, 1.2), col="blue", pch=17 )
}
#
# lasso with lars:
if (require(lars)) {
#
prostate.lasso.lars <- lars( as.matrix(train[,1:8]), train[,9],
  type="lasso", trace=TRUE )
cv.lars( as.matrix(train[,1:8]), train[,9],
  type="lasso", trace=TRUE, K=10 )
}
#
# CV (cross-validation) using package boot:
#
library(boot)
prostate.glm <- glm( lpsa ~ ., data=train )
# repeat this some times to make clear that cross-validation is
# a random procedure
#

```

```

cv.glm( train, prostate.glm, K=10 )$delta
#
# This is a two-component vector, raw cross-validated estimate and
#   adjusted cross-validated estimate.
summary( prostate.glm )
#

```

---

SAheart

*South African Heart Disease Data*


---

### Description

A retrospective sample of males in a heart-disease high-risk region of the Western Cape, South Africa.

### Usage

```
data(SAheart)
```

### Format

A data frame with 462 observations on the following 10 variables.

**sbp** systolic blood pressure

**tobacco** cumulative tobacco (kg)

**ldl** low density lipoprotein cholesterol

**adiposity** a numeric vector

**famhist** family history of heart disease, a factor with levels Absent Present

**typea** type-A behavior

**obesity** a numeric vector

**alcohol** current alcohol consumption

**age** age at onset

**chd** response, coronary heart disease

### Details

A retrospective sample of males in a heart-disease high-risk region of the Western Cape, South Africa. There are roughly two controls per case of CHD. Many of the CHD positive men have undergone blood pressure reduction treatment and other programs to reduce their risk factors after their CHD event. In some cases the measurements were made after these treatments. These data are taken from a larger dataset, described in Rousseauw et al, 1983, South African Medical Journal.

### Source

Rousseauw, J., du Plessis, J., Benade, A., Jordaan, P., Kotze, J. and Ferreira, J. (1983). Coronary risk factor screening in three rural communities, South African Medical Journal 64: 430–436.

**Examples**

```
str(SAheart)
summary(SAheart)
```

---

simple.ridge	<i>Simple Ridge Regression</i>
--------------	--------------------------------

---

**Description**

Function to do simple ridge regression.

**Usage**

```
simple.ridge(x, y, lambda = 1, df, ...)
```

**Arguments**

x	predictor matrix, NA's not allowed. Should not include a column of 1's.
y	response matrix, NA's not allowed
lambda	vector of ridge coefficients
df	vector of target degrees of freedom. lambda is calculated from these.
...	Presently not used

**Details**

Principal advantage with this compared to other implementations of ridge regression in R is that we can do the calculations for multiple degrees of freedom at once. Ridging is not applied to the intercept.

**Value**

List with components:

beta0	intercept
beta	matrix of coefficients. One column for each lambda
lambda	lambda used
df	df used

**Author(s)**

Kjetil B. Halvorsen

**Examples**

```
# See examples for \link{prostate}.
```

---

spam

*Email Spam Data*

---

### **Description**

SPAM E-mail Database. See Details below.

### **Usage**

`data(spam)`

### **Format**

A data frame with 4601 observations on the following 58 variables.

- A.1** a numeric vector
- A.2** a numeric vector
- A.3** a numeric vector
- A.4** a numeric vector
- A.5** a numeric vector
- A.6** a numeric vector
- A.7** a numeric vector
- A.8** a numeric vector
- A.9** a numeric vector
- A.10** a numeric vector
- A.11** a numeric vector
- A.12** a numeric vector
- A.13** a numeric vector
- A.14** a numeric vector
- A.15** a numeric vector
- A.16** a numeric vector
- A.17** a numeric vector
- A.18** a numeric vector
- A.19** a numeric vector
- A.20** a numeric vector
- A.21** a numeric vector
- A.22** a numeric vector
- A.23** a numeric vector
- A.24** a numeric vector
- A.25** a numeric vector

- A.26 a numeric vector
- A.27 a numeric vector
- A.28 a numeric vector
- A.29 a numeric vector
- A.30 a numeric vector
- A.31 a numeric vector
- A.32 a numeric vector
- A.33 a numeric vector
- A.34 a numeric vector
- A.35 a numeric vector
- A.36 a numeric vector
- A.37 a numeric vector
- A.38 a numeric vector
- A.39 a numeric vector
- A.40 a numeric vector
- A.41 a numeric vector
- A.42 a numeric vector
- A.43 a numeric vector
- A.44 a numeric vector
- A.45 a numeric vector
- A.46 a numeric vector
- A.47 a numeric vector
- A.48 a numeric vector
- A.49 a numeric vector
- A.50 a numeric vector
- A.51 a numeric vector
- A.52 a numeric vector
- A.53 a numeric vector
- A.54 a numeric vector
- A.55 a numeric vector
- A.56 a numeric vector
- A.57 a numeric vector
- spam** Factor w/ 2 levels "email", "spam"

## Details

The "spam" concept is diverse: advertisements for products/web sites, make money fast schemes, chain letters, pornography... Our collection of spam e-mails came from our postmaster and individuals who had filed spam. Our collection of non-spam e-mails came from filed work and personal e-mails, and hence the word 'george' and the area code '650' are indicators of non-spam. These are useful when constructing a personalized spam filter. One would either have to blind such non-spam indicators or get a very wide collection of non-spam to generate a general purpose spam filter.

For background on spam: Cranor, Lorrie F., LaMacchia, Brian A. Spam! Communications of the ACM, 41(8):74-83, 1998.

Attribute Information: The last column of 'spambase.data' denotes whether the e-mail was considered spam (1) or not (0), i.e. unsolicited commercial e-mail. Most of the attributes indicate whether a particular word or character was frequently occurring in the e-mail. The run-length attributes (55-57) measure the length of sequences of consecutive capital letters. For the statistical measures of each attribute, see the end of this file. Here are the definitions of the attributes:

48 continuous real [0,100] attributes of type word\<\_freq\<\_WORD = percentage of words in the e-mail that match WORD, i.e.  $100 * (\text{number of times the WORD appears in the e-mail}) / \text{total number of words in e-mail}$ . A "word" in this case is any string of alphanumeric characters bounded by non-alphanumeric characters or end-of-string.

6 continuous real [0,100] attributes of type char\<\_freq\<\_CHAR = percentage of characters in the e-mail that match CHAR, i.e.  $100 * (\text{number of CHAR occurrences}) / \text{total characters in e-mail}$

1 continuous real [1,...] attribute of type capital\<\_run\<\_length\<\_average = average length of uninterrupted sequences of capital letters

1 continuous integer [1,...] attribute of type capital\<\_run\<\_length\<\_longest = length of longest uninterrupted sequence of capital letters

1 continuous integer [1,...] attribute of type capital\<\_run\<\_length\<\_total = sum of length of uninterrupted sequences of capital letters = total number of capital letters in the e-mail

1 nominal {0,1} class attribute of type spam = denotes whether the e-mail was considered spam (1) or not (0), i.e. unsolicited commercial e-mail.

## Source

(a) Creators: Mark Hopkins, Erik Reeber, George Forman, Jaap Suermondt Hewlett-Packard Labs, 1501 Page Mill Rd., Palo Alto, CA 94304 (b) Donor: George Forman (gforman at nospam hpl.hp.com) 650-857-7835 (c) Generated: June-July 1999

## References

<http://www.ics.uci.edu/~mlearn/MLRepository.html>

## Examples

```
head(str(spam))
```

---

`vowel.test`*Vowel Recognition (Deterding data)*

---

**Description**

Speaker independent recognition of the eleven steady state vowels of British English using a specified training set of lpc derived log area ratios.

**Usage**

```
data(vowel.test)
```

**Format**

A data frame with 462 observations on the following 11 variables.

`y` a numeric vector  
`x.1` a numeric vector  
`x.2` a numeric vector  
`x.3` a numeric vector  
`x.4` a numeric vector  
`x.5` a numeric vector  
`x.6` a numeric vector  
`x.7` a numeric vector  
`x.8` a numeric vector  
`x.9` a numeric vector  
`x.10` a numeric vector

**Details**

See the Details section of the help file for [vowel.train](#).

**Source**

David Deterding (data and non-connectionist analysis) Mahesan Niranjan (first connectionist analysis) Tony Robinson (description, program, data, and results)

**References**

neural-bench@cs.cmu.edu

**Examples**

```
str(vowel.test)
```

---

`vowel.train`*Vowel Recognition (Deterding data)*

---

**Description**

Speaker independent recognition of the eleven steady state vowels of British English using a specified training set of lpc derived log area ratios.

**Usage**

```
data(vowel.train)
```

**Format**

A data frame with 528 observations on the following 11 variables.

**y** a numeric vector

**x.1** a numeric vector

**x.2** a numeric vector

**x.3** a numeric vector

**x.4** a numeric vector

**x.5** a numeric vector

**x.6** a numeric vector

**x.7** a numeric vector

**x.8** a numeric vector

**x.9** a numeric vector

**x.10** a numeric vector

**Details**

For a more detailed explanation of the problem, see the excerpt from Tony Robinson's Ph.D. thesis in the COMMENTS section. In Robinson's opinion, connectionist problems fall into two classes, the possible and the impossible. He is interested in the latter, by which he means problems that have no exact solution. Thus the problem here is not to see how fast a network can be trained (although this is important), but to maximise a less than perfect performance.

**METHODOLOGY:**

Report the number of test vowels classified correctly, (i.e. the number of occurrences when distance of the correct output to the actual output was the smallest of the set of distances from the actual output to all possible target outputs).

Though this is not the focus of Robinson's study, it would also be useful to report how long the training took (measured in pattern presentations or with a rough count of floating-point operations required) and what level of success was achieved on the training and testing data after various amounts of training. Of course, the network topology and algorithm used should be precisely described as well.

## VARIATIONS:

This benchmark is proposed to encourage the exploration of different node types. Please theorise/experiment/hack. The author (Robinson) will try to correspond by email if requested. In particular there has been some discussion recently on the use of a cross-entropy distance measure, and it would be interesting to see results for that.

## RESULTS:

Here is a summary of results obtained by Tony Robinson. A more complete explanation of this data is given in the excerpt from his thesis in the COMMENTS section below.

Classifier	no. of units	no. hidden	percent correct
Single-layer perceptron	-	154	33
Multi-layer perceptron	88	234	51
Multi-layer perceptron	22	206	45
Multi-layer perceptron	11	203	44
Modified Kanerva Model	528	231	50
Modified Kanerva Model	88	197	43
Radial Basis Function	528	247	53
Radial Basis Function	88	220	48
Gaussian node network	528	252	55
Gaussian node network	88	247	53
Gaussian node network	22	250	54
Gaussian node network	11	211	47
Square node network	88	253	55
Square node network	22	236	51
Square node network	11	217	50
Nearest neighbour	-	260	56

## Notes:

1. Each of these numbers is based on a single trial with random starting weights. More trials would of course be preferable, but the computational facilities available to Robinson were limited.
2. Graphs are given in Robinson's thesis showing test-set performance vs. epoch count for some of the training runs. In most cases, performance peaks at around 250 correct, after which performance decays to different degrees. The numbers given above are final performance figures after about 3000 trials, not the peak performance obtained during the run.

## REFERENCES:

- [Deterding89] D. H. Deterding, 1989, University of Cambridge, "Speaker Normalisation for Automatic Speech Recognition", submitted for PhD.
- [NiranjanFallside88] M. Niranjan and F. Fallside, 1988, Cambridge University Engineering Department, "Neural Networks and Radial Basis Functions in Classifying Static Speech Patterns", CUED/F-INFENG/TR.22.

[RenalsRohwer89-ijcnn] Steve Renals and Richard Rohwer, "Phoneme Classification Experiments Using Radial Basis Functions", Submitted to the International Joint Conference on Neural Networks, Washington, 1989.

[RabinerSchafer78] L. R. Rabiner and R. W. Schafer, Englewood Cliffs, New Jersey, 1978, Prentice Hall, "Digital Processing of Speech Signals".

[PragerFallside88] R. W. Prager and F. Fallside, 1988, Cambridge University Engineering Department, "The Modified Kanerva Model for Automatic Speech Recognition", CUED/F-INFENG/TR.6.

[BroomheadLowe88] D. Broomhead and D. Lowe, 1988, Royal Signals and Radar Establishment, Malvern, "Multi-variable Interpolation and Adaptive Networks", RSRE memo, \#4148.

[RobinsonNiranjanFallside88-tr] A. J. Robinson and M. Niranjan and F. Fallside, 1988, Cambridge University Engineering Department, "Generalising the Nodes of the Error Propagation Network", CUED/F-INFENG/TR.25.

[Robinson89] A. J. Robinson, 1989, Cambridge University Engineering Department, "Dynamic Error Propagation Networks".

[McCullochAinsworth88] N. McCulloch and W. A. Ainsworth, Proceedings of Speech'88, Edinburgh, 1988, "Speaker Independent Vowel Recognition using a Multi-Layer Perceptron".

[RobinsonFallside88-neuro] A. J. Robinson and F. Fallside, 1988, Proceedings of nEuro'88, Paris, June, "A Dynamic Connectionist Model for Phoneme Recognition.

#### COMMENTS:

(By Tony Robinson)

The program supplied is slow. I ran it on several MicroVaxII's for many nights. I suspect that if I had spent more time on it, it would have been possible to get better results. Indeed, my later code has a slightly better adaptive step size algorithm, but the old version is given here for comatability with the stated performance values. It is interesting that, for this problem, the nearest neighbour clasification outperforms any of the connectionist models. This can be seen as a challange to improve the connectionist performance.

The following problem description results and discussion is taken from my PhD thesis. The aim was to demonstrate that many types of node can be trained using gradient descent. The full thesis will be available from me when it has been examined, say maybe July 1989.

#### Application to Vowel Recognition —————

This chapter describes the application of a variety of feed-forward networks to the task of recognition of vowel sounds from multiple speakers. Single speaker vowel recognition studies by Renals and Rohwer [RenalsRohwer89-ijcnn] show that feed-forward networks compare favourably with vector-quantised hidden Markov models. The vowel data used in this chapter was collected by Deterding [Deterding89], who recorded examples of the eleven steady state vowels of English spoken by fifteen speakers for a speaker normalisation study. A range of node types are used, as described in the previous section, and some of the problems of the error propagation algorithm are discussed.

#### The Speech Data

(An ascii approximation to) the International Phonetic Association (I.P.A.) symbol and the word in which the eleven vowel sounds were recorded is given in table 4.1. The word was uttered once by each of the fifteen speakers. Four male and four female speakers were used to train the networks, and the other four male and three female speakers were used for testing the performance.

vowel	word	vowel	word
i	heed	0	hod
I	hid	C:	hoard
E	head	U	hood
A	had	u:	who'd
a:	hard	3:	heard
Y	hud		

Table 4.1: Words used in Recording the Vowels

#### Front End Analysis

The speech signals were low pass filtered at 4.7kHz and then digitised to 12 bits with a 10kHz sampling rate. Twelfth order linear predictive analysis was carried out on six 512 sample Hamming windowed segments from the steady part of the vowel. The reflection coefficients were used to calculate 10 log area parameters, giving a 10 dimensional input space. For a general introduction to speech processing and an explanation of this technique see Rabiner and Schafer [RabinerSchafer78].

Each speaker thus yielded six frames of speech from eleven vowels. This gave 528 frames from the eight speakers used to train the networks and 462 frames from the seven speakers used to test the networks.

#### Details of the Models

All the models had common structure of one layer of hidden units and two layers of weights. Some of the models used fixed weights in the first layer to perform a dimensionality expansion [Robinson89:sect3.1], and the remainder modified the first layer of weights using the error propagation algorithm for general nodes described in [Robinson89:chap2]. In the second layer the hidden units were mapped onto the outputs using the conventional weighted-sum type nodes with a linear activation function. When Gaussian nodes were used the range of influence of the nodes,  $w_{ij1}$ , was set to the standard deviation of the training data for the appropriate input dimension. If the locations of these nodes,  $w_{ij0}$ , are placed randomly, then the model behaves like a continuous version of the modified Kanerva model [PragerFallside88]. If the locations are placed at the points defined by the input examples then the model implements a radial basis function [BroomheadLowe88]. The first layer of weights remains constant in both of these models, but can be also trained using the equations of [Robinson89:sect2.4]. Replacing the Gaussian nodes with the conventional type gives a multilayer perceptron and replacing them with conventional nodes with the activation function  $f(x) = x^2$  gives a network of square nodes. Finally, dispensing with the first layer altogether yields a single layer perceptron.

The scaling factor between gradient of the energy and the change made to the weights (the 'learning rate', 'eta') was dynamically varied during training, as described in [Robinson89:sect2.5]. If the energy decreased this factor was increased by 5%, if it increased the factor was halved. The networks changed the weights in the direction of steepest descent which is susceptible to finding a local minimum. A 'momentum' term [RumelhartHintonWilliams86] is often used with error propagation networks to smooth the weight changes and 'ride over' small local minima. However, the optimal value of this term is likely to be problem dependent, and in order to provide a uniform framework, this additional term was not used.

#### Recognition Results

This experiment was originally carried out with only two frames of data from each word [RobinsonNiranjanFallside88-tr]. In the earlier experiment some problems were encountered with a phenomena termed ‘over-training’ whereby the recognition rate on the test data peaks part way through training then decays significantly. The recognition rates for the six frames per word case are given in table 4.2 and are generally higher and show less variability than the previously presented results. However, the recognition rate on the test set still displays large fluctuations during training, as shown by the plots in [Robinson89:fig3.2] Some fluctuations will arise from the fact that the minimum in weight space for the training set will not be coincident with the minima for the test set. Thus, half the possible trajectories during learning will approach the test set minimum and then move away from it again on the way to the training set minima [Mark Plumbley, personal communication]. In addition, continued training sharpens the class boundaries which makes the energy insensitive to the class boundary position [Mahesan Niranjan, personal communiation]. For example, there are a large number planes defined with threshold units which will separate two points in the input space, but only one least squares solution for the case of linear units.

Classifier	no. of units	no. hidden	percent correct	correct
Single-layer perceptron	-	154	33	
Multi-layer perceptron	88	234	51	
Multi-layer perceptron	22	206	45	
Multi-layer perceptron	11	203	44	
Modified Kanerva Model	528	231	50	
Modified Kanerva Model	88	197	43	
Radial Basis Function	528	247	53	
Radial Basis Function	88	220	48	
Gaussian node network	528	252	55	
Gaussian node network	88	247	53	
Gaussian node network	22	250	54	
Gaussian node network	11	211	47	
Square node network	88	253	55	
Square node network	22	236	51	
Square node network	11	217	50	
Nearest neighbour	-	260	56	

Table 4.2: Vowel classification with different non-linear classifiers

#### Discussion

From these vowel classification results it can be seen that minimising the least mean square error over a training set does not guarantee good generalisation to the test set. The best results were achieved with nearest neighbour analysis which classifies an item as the class of the closest example in the training set measured using the Euclidean distance. It is expected that the problem of overtraining could be overcome by using a larger training set taking data from more speakers. The performance of the Gaussian and square node network was generally better than that of the multi-layer perceptron. In other speech recognition problems which attempt to classify single frames of speech, such as those described by McCulloch and Ainsworth [McCullochAinsworth88] and that

of [Robinson89:chap7 and RobinsonFallside88-neuro], the nearest neighbour algorithm does not perform as well as a multilayer perceptron. It would be interesting to investigate this difference and apply a network of Gaussian or square nodes to these problems.

The initial weights to the hidden units in the Gaussian network can be given a physical interpretation in terms of matching to a template for a set of features. This gives an advantage both in shortening the training time and also because the network starts at a point in weight space near a likely solution, which avoids some possible local minima which represent poor solutions.

The results of the experiments with Gaussian and square nodes are promising. However, it has not been the aim of this chapter to show that a particular type of node is necessarily ‘better’ for error propagation networks than the weighted sum node, but that the error propagation algorithm can be applied successfully to many different types of node.

### Source

David Deterding (data and non-connectionist analysis) Mahesan Niranjan (first connectionist analysis) Tony Robinson (description, program, data, and results)

### References

neural-bench@cs.cmu.edu

### Examples

```
str(vowel.train)
```

---

waveform

*Function to simulate waveform data*

---

### Description

Simulates waveform data, as in the book page 402.

### Usage

```
waveform(n)
```

### Arguments

n                      Number of waveforms to simulate

### Details

The formulas are given in the book page 402.

### Value

Dataframe with two variables, x, a matrix with 21 columns, and a vector y with class labels.

**Author(s)**

Kjetil Halvorsen

**See Also**

[waveform.train](#) and [waveform.test](#).

---

waveform.test

*Simulated Waveform Data*

---

**Description**

This is a simulated three-class problem with 21 variables, and considered to be a difficult pattern recognition problem.

**Usage**

```
data(waveform.test)
```

**Format**

A data frame with 500 observations on the following 22 variables.

y a numeric vector

x.1 a numeric vector

x.2 a numeric vector

x.3 a numeric vector

x.4 a numeric vector

x.5 a numeric vector

x.6 a numeric vector

x.7 a numeric vector

x.8 a numeric vector

x.9 a numeric vector

x.10 a numeric vector

x.11 a numeric vector

x.12 a numeric vector

x.13 a numeric vector

x.14 a numeric vector

x.15 a numeric vector

x.16 a numeric vector

x.17 a numeric vector

x.18 a numeric vector

x.19 a numeric vector

x.20 a numeric vector

x.21 a numeric vector

**Examples**

```
str(waveform.test)
```

---

```
waveform.train
```

*Simulated Waveform Data*

---

**Description**

This is a simulated three-class problem with 21 variables, and considered to be a difficult pattern recognition problem.

**Usage**

```
data(waveform.train)
```

**Format**

A data frame with 300 observations on the following 22 variables.

**y** a numeric vector

**x.1** a numeric vector

**x.2** a numeric vector

**x.3** a numeric vector

**x.4** a numeric vector

**x.5** a numeric vector

**x.6** a numeric vector

**x.7** a numeric vector

**x.8** a numeric vector

**x.9** a numeric vector

**x.10** a numeric vector

**x.11** a numeric vector

**x.12** a numeric vector

**x.13** a numeric vector

**x.14** a numeric vector

**x.15** a numeric vector

**x.16** a numeric vector

**x.17** a numeric vector

**x.18** a numeric vector

**x.19** a numeric vector

**x.20** a numeric vector

**x.21** a numeric vector

**Examples**

```
str(waveform.train)
```

---

`zip.test`*Handwritten Digit Recognition Data*

---

**Description**

This example is a character recognition task: classification of handwritten numerals. This problem captured the attention of the machine learning and neural network community for many years, and has remained a benchmark problem in the field.

**Usage**

```
data(zip.test)
```

**Format**

The format is: num [1:2007, 1:257] 9 6 3 6 6 0 0 6 9 ...

**Details**

Normalized handwritten digits, automatically scanned from envelopes by the U.S. Postal Service. The original scanned digits are binary and of different sizes and orientations; the images here have been deslanted and size normalized, resulting in 16 x 16 grayscale images (Le Cun et al., 1990).

The data are in two gzipped files, and each line consists of the digit id (0-9) followed by the 256 grayscale values.

There are 7291 training observations and 2007 test observations, distributed as follows: 0 1 2 3 4 5 6 7 8 9 Total Train 1194 1005 731 658 652 556 664 645 542 644 7291 Test 359 264 198 166 200 160 170 147 166 177 2007

or as proportions: 0 1 2 3 4 5 6 7 8 9 Train 0.16 0.14 0.1 0.09 0.09 0.08 0.09 0.09 0.07 0.09 Test 0.18 0.13 0.1 0.08 0.10 0.08 0.08 0.07 0.08 0.09

The test set is notoriously "difficult", and a 2.5 excellent. These data were kindly made available by the neural network group at AT&T research labs (thanks to Yann Le Cunn).

---

`zip.train`*Handwritten Digit Recognition Data*

---

**Description**

This example is a character recognition task: classification of handwritten numerals. This problem captured the attention of the machine learning and neural network community for many years, and has remained a benchmark problem in the field.

**Usage**

```
data(zip.train)
```

**Format**

The format is: num [1:7291, 1:257] 6 5 4 7 3 6 3 1 0 1 ...

**Details**

Normalized handwritten digits, automatically scanned from envelopes by the U.S. Postal Service. The original scanned digits are binary and of different sizes and orientations; the images here have been deslanted and size normalized, resulting in 16 x 16 grayscale images (Le Cun et al., 1990).

The data are in two gzipped files, and each line consists of the digit id (0-9) followed by the 256 grayscale values.

There are 7291 training observations and 2007 test observations, distributed as follows: 0 1 2 3 4 5 6 7 8 9 Total Train 1194 1005 731 658 652 556 664 645 542 644 7291 Test 359 264 198 166 200 160 170 147 166 177 2007

or as proportions: 0 1 2 3 4 5 6 7 8 9 Train 0.16 0.14 0.1 0.09 0.09 0.08 0.09 0.09 0.07 0.09 Test 0.18 0.13 0.1 0.08 0.10 0.08 0.07 0.08 0.09

The test set is notoriously "difficult", and a 2.5 excellent. These data were kindly made available by the neural network group at AT&T research labs (thanks to Yann Le Cunn).

**Examples**

```
findRows <- function(zip, n) {
  # Find n (random) rows with zip representing 0,1,2,...,9
  res <- vector(length=10, mode="list")
  names(res) <- 0:9
  ind <- zip[,1]
  for (j in 0:9) {
    res[[j+1]] <- sample( which(ind==j), n ) }
  return(res) }

# Making a plot like that on page 4:

digits <- vector(length=10, mode="list")
names(digits) <- 0:9
rows <- findRows(zip.train, 6)
for (j in 0:9) {
  digits[[j+1]] <- do.call("cbind", lapply(as.list(rows[[j+1]]),
    function(x) zip2image(zip.train, x)) )
}
im <- do.call("rbind", digits)
image(im, col=gray(256:0/256), zlim=c(0,1), xlab="", ylab="" )
```

---

zip2image

*function to convert row of zip file to format used by image()*


---

**Description**

This is a utility function converting zip.train/zip.test data to format useful for plotting with the function [image](#).

**Usage**

```
zip2image(zip, line)
```

**Arguments**

zip	<a href="#">zip.train</a> or <a href="#">zip.test</a> .
line	row of matrix to take

**Value**

16 x 16 matrix suitable as argument for [image](#).

**Author(s)**

Kjetil Halvorsen

**Examples**

```
## See example section of help file for zip.train
```

# Index

## \*Topic **datagen**

waveform, [35](#)

## \*Topic **datasets**

bone, [2](#)

countries, [3](#)

galaxy, [4](#)

marketing, [5](#)

mixture.example, [6](#)

nci, [9](#)

orange10.test, [10](#)

orange10.train, [11](#)

orange4.test, [11](#)

orange4.train, [12](#)

ozone, [13](#)

phoneme, [13](#)

prostate, [21](#)

SAheart, [24](#)

spam, [26](#)

vowel.test, [29](#)

vowel.train, [30](#)

waveform.test, [36](#)

waveform.train, [37](#)

zip.test, [38](#)

zip.train, [38](#)

## \*Topic **dplot**

zip2image, [39](#)

## \*Topic **regression**

simple.ridge, [25](#)

bone, [2](#)

countries, [3](#)

galaxy, [4](#)

image, [39](#), [40](#)

marketing, [5](#)

mixture.example, [6](#)

nci, [9](#)

orange10.test, [10](#)

orange10.train, [11](#)

orange4.test, [11](#)

orange4.train, [12](#)

ozone, [13](#)

phoneme, [13](#)

prostate, [21](#)

SAheart, [24](#)

simple.ridge, [25](#)

spam, [26](#)

vowel.test, [29](#)

vowel.train, [29](#), [30](#)

waveform, [35](#)

waveform.test, [36](#), [36](#)

waveform.train, [36](#), [37](#)

zip.test, [38](#), [40](#)

zip.train, [38](#), [40](#)

zip2image, [39](#)