STAT 210
Applied Statistics and Data Analysis
Multivariate Density Estimation

Joaquin Ortega

Fall 2020

Multivariate Density Estimation

Multivariate density estimation

Kernel density estimation can be extended to estimate multivariate densities f in $\mathbb{R}^d$ using the same ideas: average densities centered at the sample points.

For a sample $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$ the estimated density evaluated at $\mathbf{x} \in \mathbb{R}^d$ is given by

$$f(\mathbf{x}; \mathbf{H}) = \frac{1}{n|\mathbf{H}|^{1/2}} \sum_{i=1}^{n} K(\mathbf{H}^{-1/2}(\mathbf{x} - \mathbf{x}_i)) \tag{1}$$

where K is a multivariate kernel, a d-variate density that is symmetric and usually unimodal at $\mathbf{0}$ and $\mathbf{H}$ is the bandwidth matrix, a $d \times d$ positive-definite matrix.

Multivariate density estimation

Notation:

$$K_{\mathbf{H}}(\mathbf{x}) = \frac{1}{|\mathbf{H}|^{1/2}} K(\mathbf{H}^{-1/2}\mathbf{x}) \tag{2}$$

We will consider only the multivariate normal kernel, which is the most commonly used.

In this case, we can think of the bandwidth matrix as the variance-covariance matrix of a multivariate normal density whose mean is $\mathbf{x}_i$.

Multivariate density estimation

The simplest case is when the coordinates are independent, and the multivariate kernel is the product of univariate Gaussian densities.

In the two-dimensional case, we write this as

$$\hat{f}(x, y; \mathbf{h}) = \frac{1}{n} \sum_{i=1}^{n} K_{h_1}(x - x_i) K_{h_2}(y - y_i) \tag{3}$$

where $\mathbf{h} = (h_1, h_2)$ is the bandwidth and $K(x)$ is a standard Gaussian kernel, but other kernels are also possible.

Multivariate density estimation

As an example, we first generate a gaussian sample using the package mvtnorm.

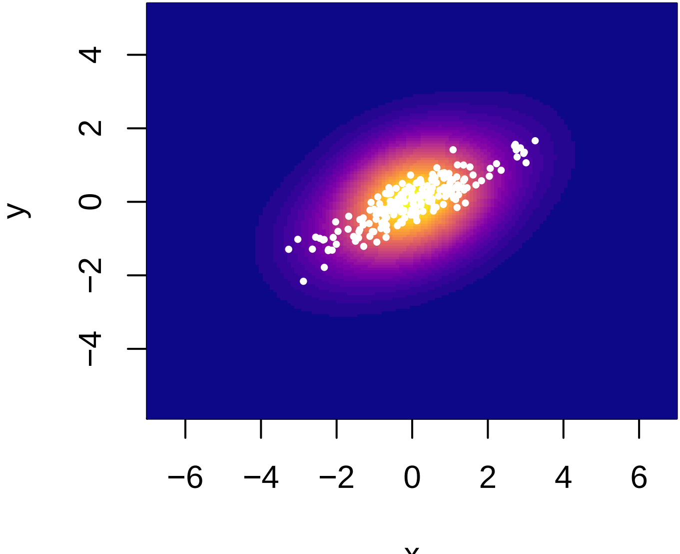
```
set.seed(2345)
x <- mvtnorm::rmvnorm(200,sigma =
                rbind(c(1.5,0.75),c(0.75,0.5)))
```

We use a diagonal matrix with ones as the bandwidth matrix.

```
H <- diag(c(1,1))
kde <- ks::kde(x, H)
```

Multivariate density estimation

```
image(kde$eval.points[[1]], kde$eval.points[[2]], kde$estimate,
      col = viridis::plasma(30), xlab = "x", ylab = "y")
points(kde$x, cex=.5, col = 'white', pch = 16)
```



Multivariate density estimation

In the multivariate case, it is also possible to evaluate the density at given points.

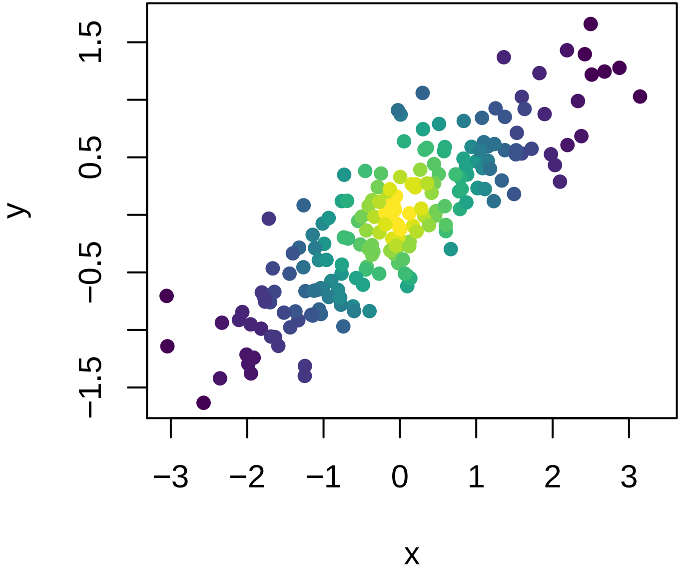
In this example, we generate a sample of 200 points from the same Gaussian distribution and then evaluate the density in them. The density value is depicted in a color scale.

Multivariate density estimation

```
pts <- mvtnorm::rmvnorm(200
                sigma = rbind(c(1.5,0.75),c(0.75,0.5)))
kde_sample <- ks::kde(x = x, H = H, eval.points = pts)

n_cols <- 20
quantiles <- quantile(kde_sample$estimate,
                      probs = seq(0, 1, l = n_cols + 1))
col <- viridis::viridis(n_cols)[cut(kde_sample$estimate,
                                    breaks = quantiles)]
plot(pts, col = col, pch = 19, xlab = "x", ylab = "y")
```

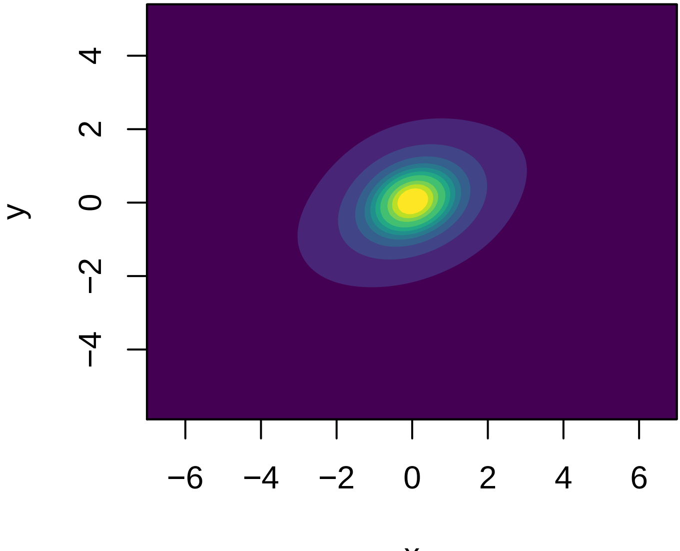
Multivariate density estimation



Multivariate density estimation

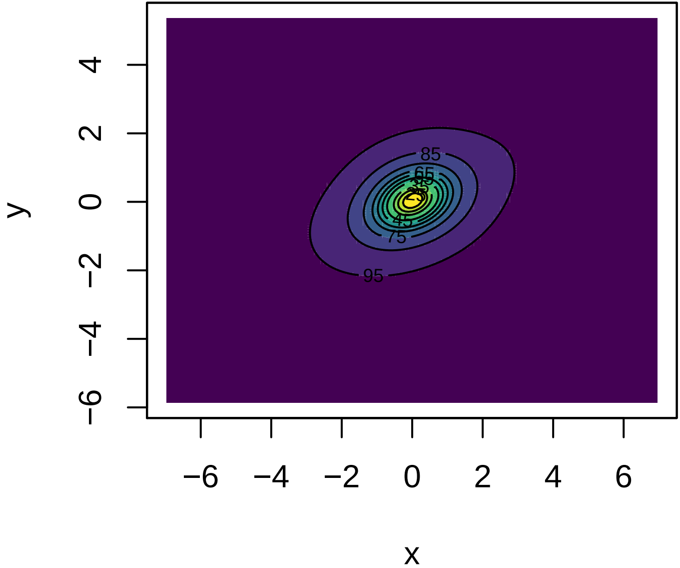
Other possible representations for the density

```
plot(kde, display = "filled.contour2", cont = seq(5, 95, by = 10),
     xlab = "x", ylab = "y", col.fun = viridis::viridis)
```



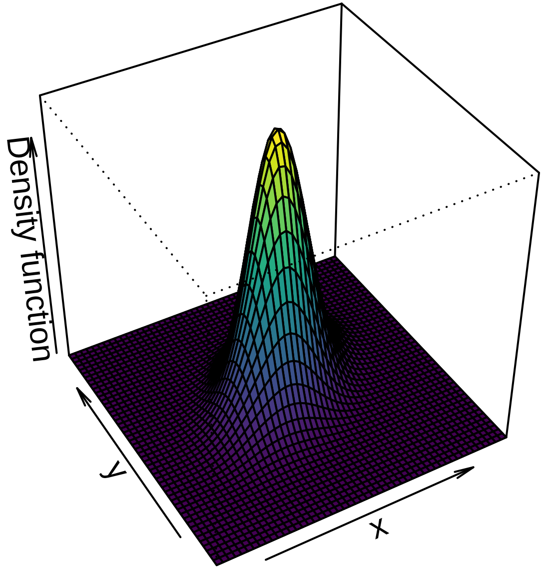
Multivariate density estimation

```
plot(kde, display = "filled.contour", cont = seq(5, 95, by = 10),
     xlab = "x", ylab = "y", col.fun = viridis::viridis)
plot(kde, display = "slice", cont = seq(5, 95, by = 10), add = TRUE)
```



Multivariate density estimation

```
plot(kde, display = "persp", col.fun = viridis::viridis,
     xlab = "x", ylab = "y")
```



Multivariate density estimation

Simulated Trivariate normal distribution

```
library(rgl)
n <- 500
set.seed(213212)
x <- mvtnorm::rmvnorm(n = n, mean = c(0, 0, 0),
                      sigma = rbind(c(1.5, 0.25, 0.5),
                                    c(0.25, 0.75, 1),
                                    c(0.5, 1, 2)))

# Show nested contours of high density regions
plot(ks::kde(x = x, H = diag(c(rep(1.25, 3)))),
     drawpoints = TRUE, col.pt = 1)
rgl::rglwidget()
```

Example: geyser data

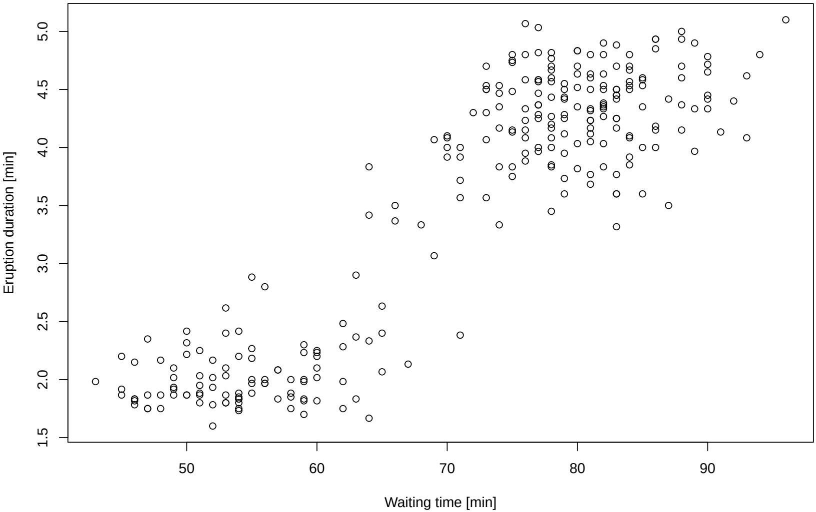
As an example, let us consider the data set `faithful`, which has data on waiting time between eruptions and the duration of the eruption, both in minutes, for the Old Faithful geyser in Yellowstone National Park, Wyoming, USA.

```
str(faithful)
```

```
## 'data.frame':    272 obs. of  2 variables:
##  $ eruptions: num  3.6 1.8 3.33 2.28 4.53 ...
##  $ waiting  : num  79 54 74 62 85 55 88 85 51 85 ...
```

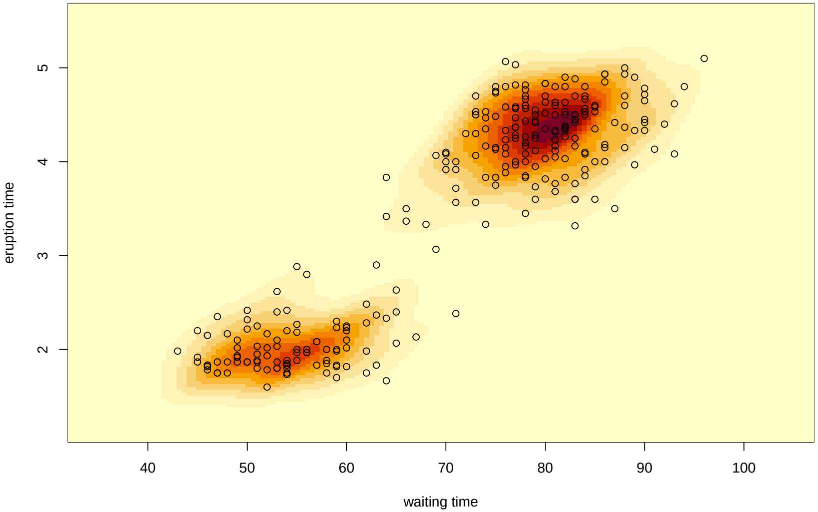
Example: geyser data

```
plot(eruptions ~ waiting, data = faithful,
     xlab = "Waiting time [min]", ylab = "Eruption duratio
```

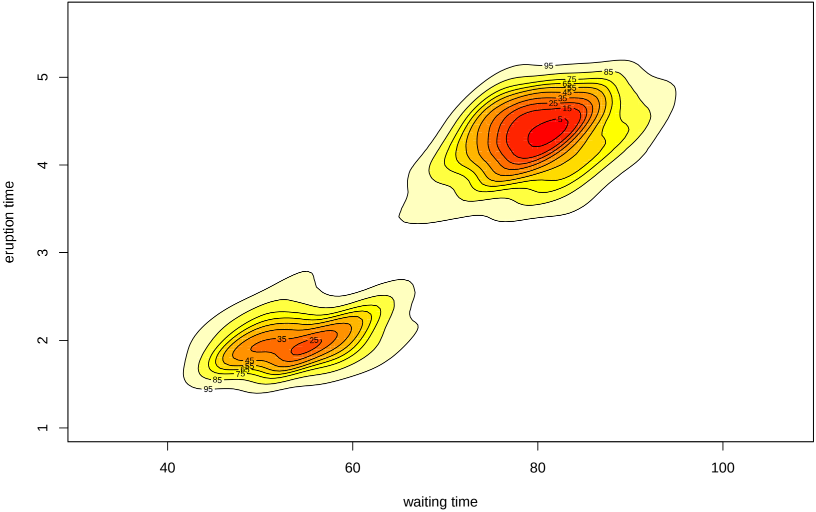


Example: geyser data

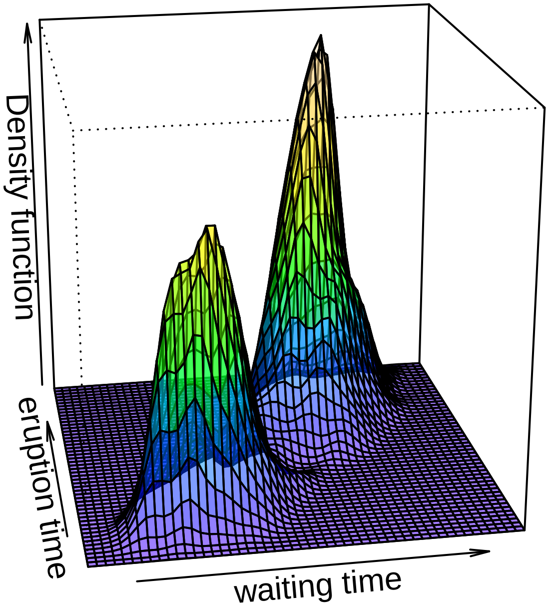
Based on this sample, let us estimate the joint density for these two variables.



Example: geyser data



Example: geyser data



Applications of Density Estimation

Classification

Consider a certain population that is divided into m groups and assume we have a sample

$$(\mathbf{X}_1, Y_1), (\mathbf{X}_2, Y_2), \ldots, (\mathbf{X}_n, Y_n)$$

where Y_i is a label indicating to which group the point $\mathbf{X}_i$ belongs.

Denote by f_j the conditional probability density function of $\mathbf{X}|Y = j$ and by π_j the probability $P(Y = j)$ for $j = 1, \ldots, m$.

In this context, we can use the law of total probability to obtain the unconditional density f of $\mathbf{X}$:

$$f(\mathbf{x}) = \sum_{j=1}^{m} f_j(\mathbf{x})\pi_j.$$

Classification

In the classification problem, we want to assign an observation $\mathbf{x}$ of $\mathbf{X}$ to one of the m classes represented by the variable Y.

A possible solution for this problem is to use the conditional probability

$$P(Y = j|\mathbf{X} = \mathbf{x}) = \frac{f_j(\mathbf{x})P(Y = j)}{f(\mathbf{x})} = \frac{\pi_j f_j(\mathbf{x})}{\sum_{j=1}^{m} f_j(\mathbf{x})\pi_j}. \tag{4}$$

The Bayes classifier assigns $\mathbf{x}$ to the most likely class, i.e., the class with the highest conditional probability.

Classification

We define the Bayes classifier as

$$B(\mathbf{x}) = \arg\max_j \pi_j f_j(\mathbf{x}).$$

The Bayes classifier has the minimum error rate among all possible classifiers but, since it depends on unknown densities f_i and probabilities π_i for $i = 1, \ldots, n$, it cannot be computed.

A possible (approximate) solution is to estimate the unknown densities using the sample $(\mathbf{X}_1, Y_1), (\mathbf{X}_2, Y_2), \ldots, (\mathbf{X}_n, Y_n)$ and use them instead of the population densities.

Classification

To estimate π_j use the relative frequencies

$$\hat{\pi}_j = \frac{n_j}{n},$$

where n_j is the number of point in the sample that belong to class j.

To estimate f_j use a kernel density estimator with the subsample of points that belong to class j: $\{\mathbf{X}_i : Y_i = j\}$. Denote this estimator by $\hat{f}_j(\cdot) = \hat{f}_j(\cdot; \mathbf{H})$.

Plugging-in these estimates into the definition of the Bayes classifier gives

$$\hat{B}(\mathbf{x}; \mathbf{H}_1, \ldots, \mathbf{H}_m) = \arg \max_{j=1,\ldots,m} \hat{\pi}_j \hat{f}_j(\mathbf{x}; \mathbf{H}_j).$$

This method is known as **kernel discriminant analysis**.

Classification

The function `kda` in package `ks` implements kernel discriminant analysis.

We illustrate the usage of this function for three groups using the `iris` dataset.

```
library(ks)
x <- iris$Sepal.Width
groups <- iris$Species
kda1 <- kda(x = x, x.group = groups)
kda1$prior.prob
```

```
## [1] 0.3333333 0.3333333 0.3333333
```

Classification

```r
head(kda1$x.group.estimate)
```

```
## [1] setosa    virginica setosa    virginica setosa
## [6] setosa
## Levels: setosa versicolor virginica
```

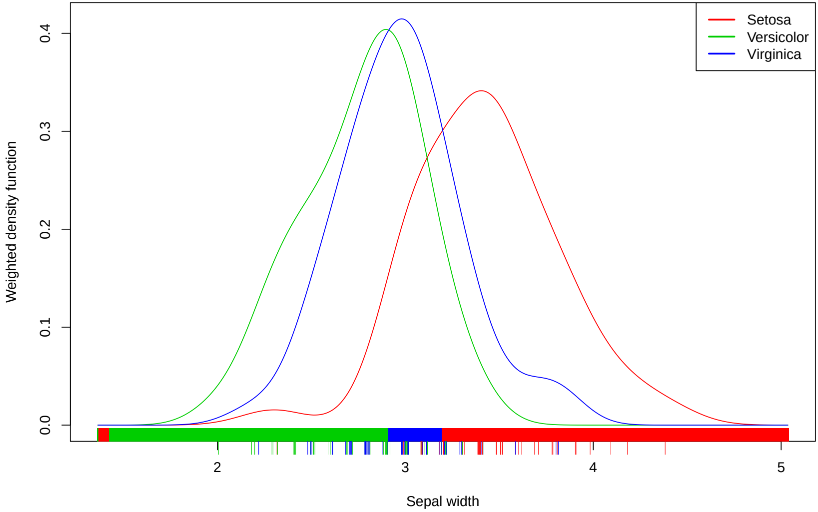
Classification

```r
options(width=80)
compare(x.group = kda1$x.group,
        est.group = kda1$x.group.estimate)
```

```
## $cross
##                    setosa (est.) versicolor (est.) virginica (est.) Total
## setosa (true)                 38                 2               10    50
## versicolor (true)              5                34               11    50
## virginica (true)              13                21               16    50
## Total                         56                57               37   150
##
## $error
## [1] 0.4133333
```

Classification

```
plot(kda1, xlab = "Sepal width", drawpoints = TRUE, col = 2:4)
legend("topright", legend = c("Setosa", "Versicolor", "Virginica"),
       lwd = 2, col = 2:4)
```



Classification

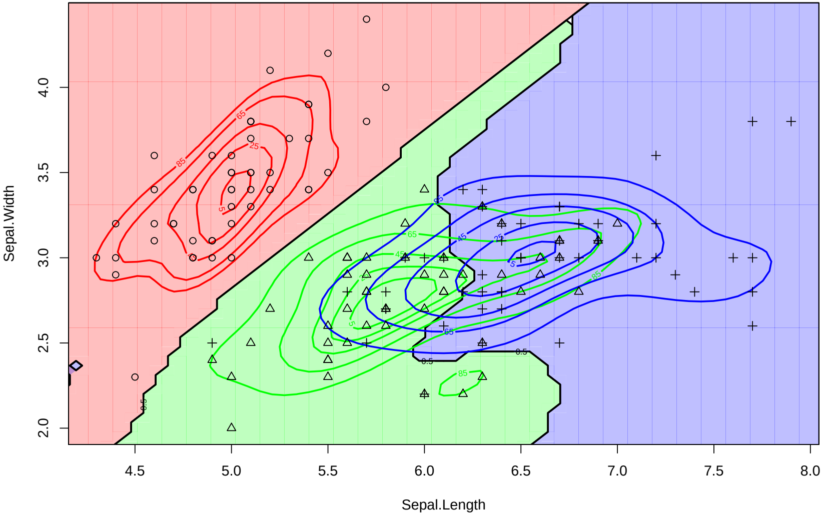
Bivariate example.

```
x <- iris[, 1:2]
groups <- iris$Species
kda2 <- kda(x = x, x.group = groups, supp = 10)
compare(x.group = kda2$x.group,
        est.group = kda2$x.group.estimate)
```

```
## $cross
##                      setosa (est.) versicolor (est.) virginica (est.) Total
## setosa (true)                   50                 0                0    50
## versicolor (true)                0                37               13    50
## virginica (true)                 0                11               39    50
## Total                           50                48               52   150
##
## $error
## [1] 0.16
```

Classification

```r
plot(kda2, col = rainbow(3), lwd = 2, col.pt = 1, cont = seq(5, 85, by = 20),
     col.part = rainbow(3, alpha = 0.25), drawpoints = TRUE)
```



Classification

Trivariate example

```
x <- iris[, 1:3]
groups <- iris$Species
# Normal scale bandwidths to avoid undersmoothing
Hs <- rbind(Hns(x = x[groups == "setosa", ]),
            Hns(x = x[groups == "versicolor", ]),
            Hns(x = x[groups == "virginica", ]))
kda3 <- kda(x = x, x.group = groups, Hs = Hs)
compare(x.group = kda3$x.group,
        est.group = kda3$x.group.estimate)
```

```
## $cross
##                    setosa (est.) versicolor (est.) virginica (est.) Total
## setosa (true)                 50                 0                0    50
## versicolor (true)              0                48                2    50
## virginica (true)               0                 1               49    50
## Total                         50                49               51   150
##
## $error
## [1] 0.02
```

Classification

Classification regions

```
plot(kda3, drawpoints = TRUE, col.pt = c(2, 3, 4),
     cont = seq(5, 85, by = 20))
rgl::rglwidget()
```