Министерство образования и науки Российской Федерации

Калужский филиал
федерального государственного бюджетного образовательного
учреждения высшего образования
**«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»**
(КФ МГТУ им. Н.Э. Баумана)

И.И. Кручинин
(к.т.н. доцент)

ЛАБОРАТОРНАЯ РАБОТА № 6
по курсу «Методы машинного обучения»

Методы классификации многомерных объектов пересекающихся классов
с использованием карт Кохонена

Калуга
2018

Теоретические основы.

Самоорганизующиеся карты Кохонена

Самоорганизующиеся карты (SOM, Self Organizing Maps), разработанные Т. Кохоненом (Kohonen, 1982), представляют собой мощный инструмент, объединяющий две важные парадигмы анализа данных – кластеризацию и проецирование, т.е. визуализацию многомерных данных на плоскости. Процедура настройки SOM относится к алгоритмами обучения без учителя.

Сеть Кохонена имеет всего два слоя: входной и выходной, составленный из радиальных нейронов упорядоченной структуры (выходной слой называют также слоем топологической карты, или “экраном”). Нейроны выходного слоя располагаются в узлах двумерной сетки с прямоугольными или шестиугольными ячейками. Количество нейронов в сетке p определяет степень детализации результата работы алгоритма, и, в конечном счете, от этого зависит точность обобщающей способности карты.
Самоорганизующиеся карты в ходе своего обучения анализируют характер расположения точек входного слоя в m-мерном пространстве и стремятся воспроизвести на выходе нейронной сети топологический порядок и определенную степень регулярности исходных данных (т.е. метрическую близость векторов). Подгонка SOM заключается в итеративной настройке вектора весовых коэффициентов wj каждого нейрона, j=1,2…,p для чего используется модифицированный алгоритм соревновательного обучения Хебба, который учитывает не только вклад нейрона-победителя, но и ближайших его соседей, расположенных в R -окрестности:

1. На стадии инициализации всем весовым коэффициентам присваиваются небольшие случайные значения w0ij,i=1,2,…,m .
2. На выходы сети подаются последовательно в случайном порядке образы y объектов входного слоя и для каждого из них выбирается “нейрон-победитель” (BMU, Best Matching Unit) с минимальным расстоянием ∑mi=1(yi−wtij) Определяется подмножество “ближайшего окружения” BMU, радиус которого R уменьшается с каждой итерацией t .
3. Пересчитываются веса wtj выделенных узлов с учетом их расстояний до нейрона-победителя и близости к вектору y .

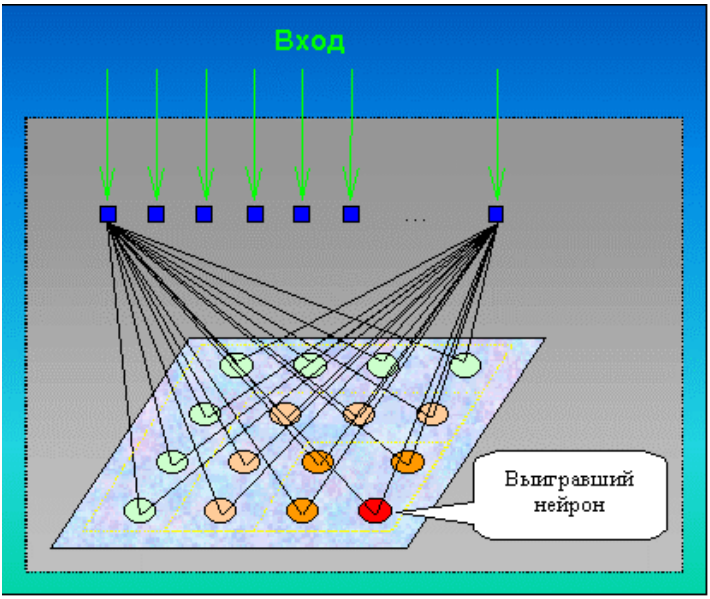


Схема активации нейронов в сети Кохонена

Шаги 2-4 алгоритма повторяются, пока выходные значения сети не будут стабилизированы с заданной точностью. При этом качество проецирования многомерных данных на плоскость достигается в SOM на нескольких уровнях: *сохранение топологии* (т.е. на множествах точек исходных данных и нейронов обученной сети структура соседства одинакова), *сохранение порядка* (т.е. расстояния между эквивалентными парами точек пропорциональны) и *сохранение метрических свойств при сжатии пространства*.

“Проекционный экран” в результате обучения приобретает свойства упорядоченной структуры, в которой величины синапсов нейронов плавно меняются вдоль двух измерений. Цвет и расположение фрагментов двумерной решетки используется для анализа закономерностей, связываемых с компонентами набора данных. В частности, с каждым узлом (нейроном) могут ассоциироваться локальные сгущения исходных объектов, которые могут служить потенциальными центрами кластеров.

Для обучения сети обычно используются функции somgrid() и som() из пакета kohonen. По завершении итерационного процесса функции plot() становится доступным для визуализации следующий комплект карт:

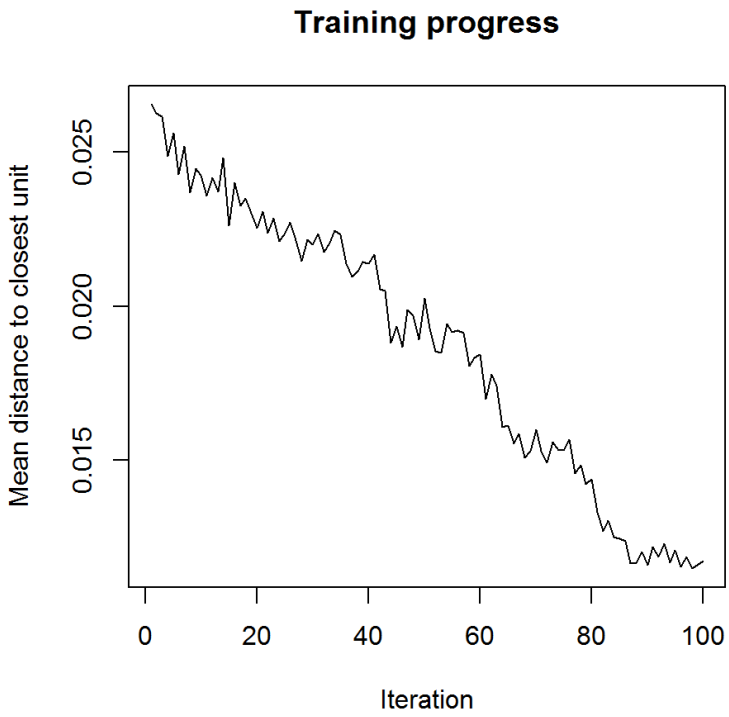
- "codes" - показывается распределение по решетке соотношение долей участия отдельных исходных переменных;
- "counts" - число исходных объектов в каждом узле сети;
- "mapping" - координаты исходных объектов на сформированной карте;
- "property", "quality", "dist.neighbours" - различными цветами изображается целый набор свойств каждого узла: доли участия отдельных исходных переменных, меры парных или средних расстояний между нейронами и т.д.

Опять воспользуемся в качестве примера набором Boston из пакета MASS. Чтобы график "codes" был более лаконичен, из исходного набора признаков выделим 7 переменных, предположительно наиболее значимых для кластеризациию. Выполним предварительно стандартизацию данных.

Укажем функции somgrid() создать для проекционного экрана гексагональную решетку 6×9, т.е. 506 земельных участков Бостона будут “самоорганизовываться” на 54 нейронах выходного слоя:

```
data(Boston, package = "MASS")
VarName = c("indus", "dis", "nox",  "medv", "lstat", "age", "rad")
# отбор переменных для обучения SOM
data_train <- Boston[, VarName]
data_train_matrix <- as.matrix(scale(data_train))

library(kohonen)
set.seed(123)
som_grid <- somgrid(xdim = 9, ydim = 6, topo = "hexagonal")
som_model <- som(data_train_matrix, grid = som_grid, rlen = 100,
                 alpha = c(0.05,0.01), keep.data = TRUE)
plot(som_model, type = "changes")
```



Снижение среднего расстояния до ближайших нейронов в ходе 100 итераций (rlen) обучения сети SOM при заданных значениях гипер-параметра alpha

Выполним теперь визуализацию комплекта карт Кохонена с разными управляющими параметрами функции plot().

```
# Зададим палитру цветов
coolBlueHotRed <- function(n, alpha = 1) {
   rainbow(n, end = 4/6, alpha = alpha)[n:1]
}
par(mfrow = c(2, 1))
# Сколько объектов связано с каждым узлом?
plot(som_model, type = "counts", palette.name = coolBlueHotRed)
```

# Каково среднее расстояние объектов узла до его прототипов?
plot(som_model, type = "quality", palette.name = coolBlueHotRed)

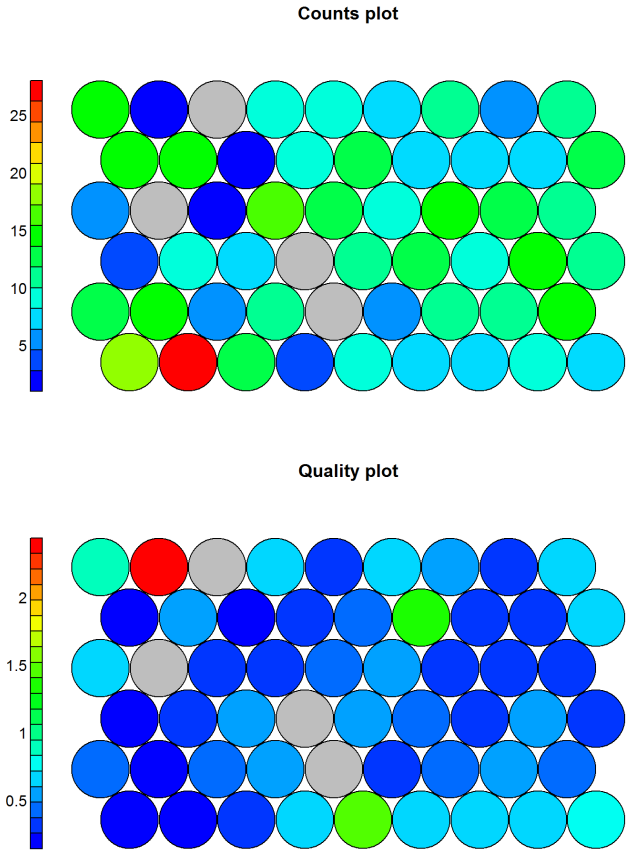


Рисунок 10.21: Карты SOM типа "counts" и "quality"

Тип карты "mapping" позволяет получить распределение объектов по узлам, удовлетворяющее любому заданному условию. Например, мы желаем выделить участки с низкой долей афроамериканцев (black - признак, который в настройке сети не участвовал). А также показать, как при этом распределяются доли участия отдельных исходных переменных:

```
colB <- ifelse(Boston$black <= 100, "red", "gray70")
par(mfrow = c(2, 1))
plot(som_model, type = "mapping", col = colB, pch = 16)
plot(som_model, type = "codes")
```

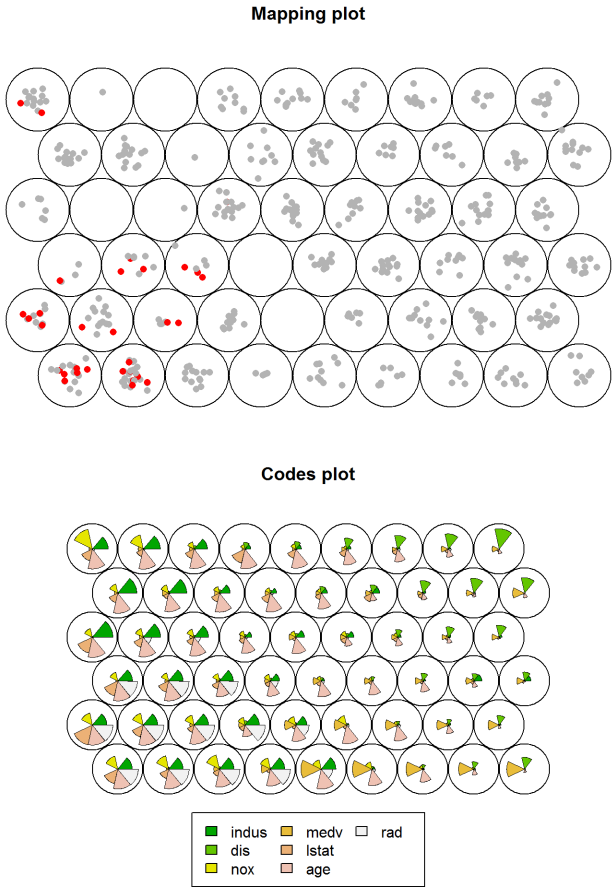
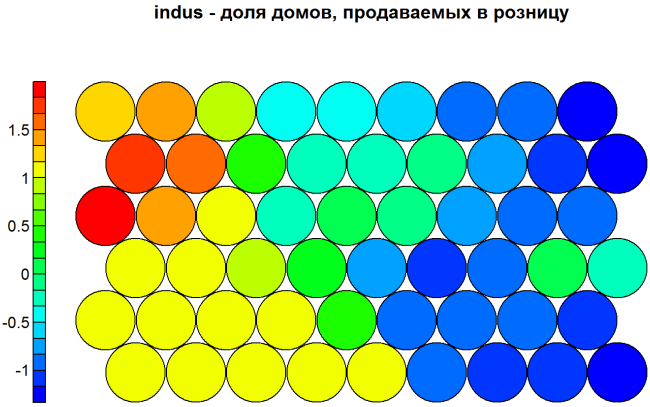
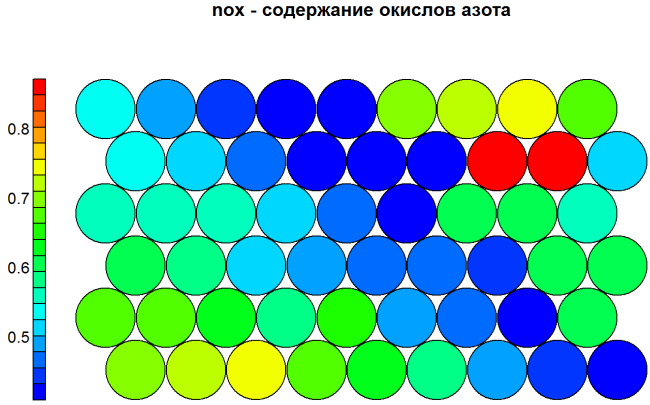


Рисунок: Карты SOM типа "mapping" и "codes"

Поскольку объединенная карта "codes" не всегда бывает хорошо интерпретируемой, можно получить карту распределения любого показателя в его стандартизованной или натуральной шкале:

```
par(mfrow = c(2, 1))
plot(som_model, type = "property",
    property = som_model$codes[[1]][,1],
    main = "indus - доля домов, продаваемых в розницу",
    palette.name = coolBlueHotRed)
var_unscaled <- aggregate(as.numeric(data_train[, 3]),
                          by = list(som_model$unit.classif),
                          FUN = mean, simplify = TRUE)[, 2]
plot(som_model, type = "property", property = var_unscaled,
    main = "nox - содержание окислов азота",
    palette.name = coolBlueHotRed)
```

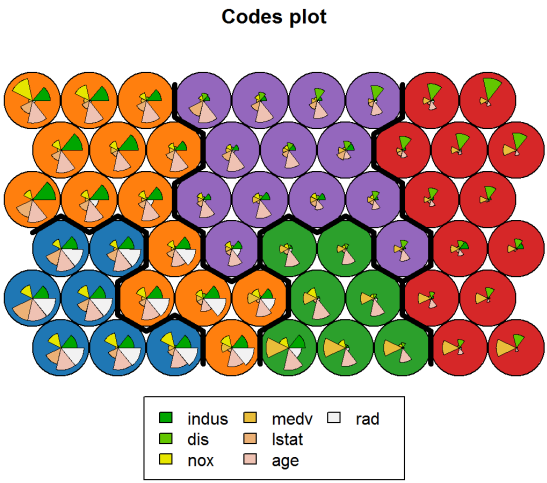




Карты SOM для стандартизованного и исходного показателя

Естественно, что значения активации каждого нейрона по каждому предиктору можно использовать для группировки узлов. Зададимся числом кластеров k=5 и выполним иерархическую кластеризацию (по умолчанию используются method = "complete" и distance = "euclidean"). Можно построить карты типа "mapping" (с метками объектов) или "codes" (с распределением доли вклада переменных). Остановимся на втором типе:

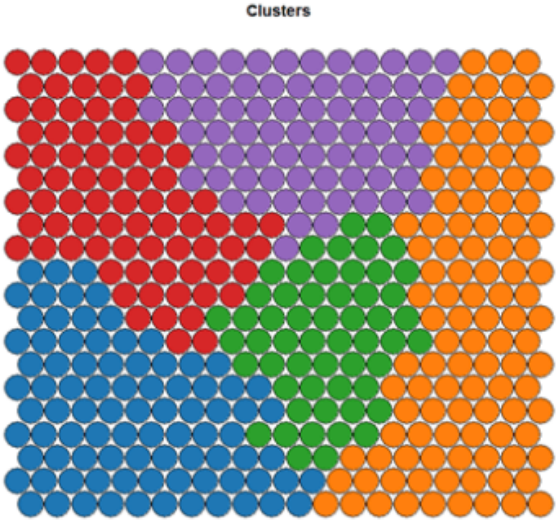
```
## Формируем матрицу "узлы ✐ переменные"
mydata <- as.matrix(som_model$codes[[1]])
# Используем иерархическую кластеризацию с порогом при k=5
som_cluster <- cutree(hclust(dist(mydata)), 5)
# Определяем палитру цветов
pretty_palette <- c("#1f77b4", '#ff7f0e', '#2ca02c',
                    '#d62728', '#9467bd', '#8c564b', '#e377c2')
# Показываем разными цветами кластеры узлов и переменные
plot(som_model, type = "codes",
     bgcol = pretty_palette[som_cluster])
add.cluster.boundaries(som_model, som_cluster)
```



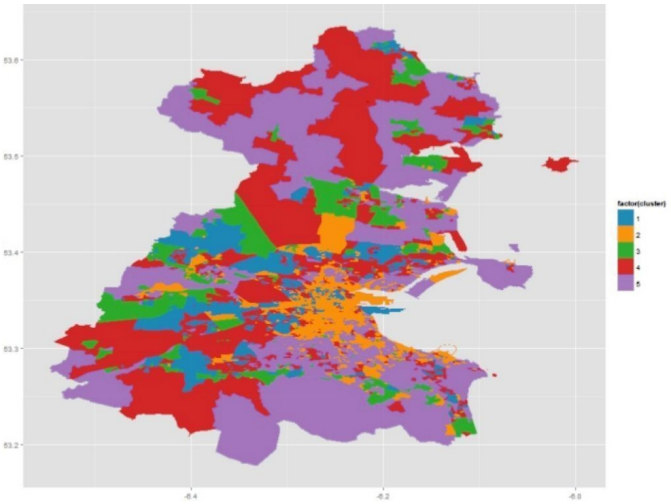
Кластеризация узлов карты SOM

Традиционно метод Кохонена рассматривается как эмпирический алгоритм, а выводы (в первую очередь, качественные) о структуре данных делаются на основе визуального анализа представленных карт. Основная трудность применения SOM, как и в случае анализа главных компонент, заключается в смысловой интерпретации топологии сети и связывании ее отдельных участков с некоторыми конкретными обобщениями из предметной области.

Однако алгоритм SOM нашел широкое применение в ГИС-технологиях, поскольку легко реализовать информационную цепочку от исходной таблицы к узлам решетки, а от них - к конкретным координатам на географической карте. Например, участники коллектива “Dublin R Users Group” использовали результаты переписи населения 2011 г., разбили территорию Дублина на 18500 маленьких площадок и описали проживающее на них население с использованием 767 переменных из



15 разделов.



Кластерная форма карты SOM и ее отображение на карте Дублина (категории от 1 до 6 связываются с градациями жилой застройки и обеспеченности населения - от трущоб до благополучных элитных районов).

Пример практической реализации карты Кохонена на языке R приведен ниже:

```
# Load the kohonen package require(kohonen)

# Create a training data set (rows are samples, columns are variables
# Here I am selecting a subset of my variables available in "data"
data_train <- data[, c(2,4,5,8)]

# Change the data frame with training data to a matrix
# Also center and scale all variables to give them equal importance during
# the SOM training process.
data_train_matrix <- as.matrix(scale(data_train))

# Create the SOM Grid - you generally have to specify the size of the
# training grid prior to training the SOM. Hexagonal and Circular
# topologies are possible
som_grid <- somgrid(xdim = 20, ydim=20, topo="hexagonal")

# Finally, train the SOM, options for the number of iterations,
# the learning rates, and the neighbourhood are available
som_model <- som(data_train_matrix,
                 grid=som_grid,
                 rlen=100,
alpha=c(0.05,0.01),
                 keep.data =
TRUE,
n.hood="circular" )



plot(som_model, type="changes")
plot(som_model, type="count")
plot(som_model, type="dist.neighbours")
plot(som_model, type="codes")

plot(som_model, type = "property", property = som_model$codes[,4],
main=names(som_model$data)[4], palette.name=coolBlueHotRed)

var <- 2 #define the variable to plot
```

```
var_unscaled <- aggregate(as.numeric(data_train[,var]),
by=list(som_model$unit.classif), FUN=mean, simplify=TRUE)[,2]
plot(som_model, type = "property", property=var_unscaled,
main=names(data_train)[var], palette.name=coolBlueHotRed)

var <- 2 #define the variable to plot
var_unscaled <- aggregate(as.numeric(data_train[,var]),
by=list(som_model$unit.classif), FUN=mean, simplify=TRUE)[,2]
plot(som_model, type = "property", property=var_unscaled,
main=names(data_train)[var], palette.name=coolBlueHotRed)
```

RBF – сети.

Также к алгоритмам классификации основанным на нейросетевой теории можно отнести сети с радиальными базисными функциями RBF. Обучение RBF-сети сводится к восстановлению плотностей классов $p_y(x)$ с помощью EM-алгоритма. Результатом обучения являются центры μ_{yj} и дисперсии Σ_{yj} компонент $j = 1,...,k_y$. Оценивая дисперсии, мы фактически подбираем веса признаков в метриках $\rho_{yj}(x,\mu_{yj})$ для каждого центра μ_{yj}.

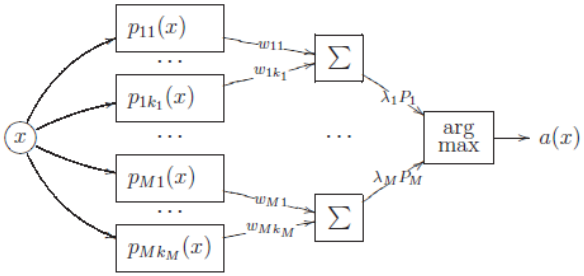


Рис. 5. Сеть радиальных базисных функций представляет собой трёхуровневую суперпозицию.

Пусть функции правдоподобия классов $p_y(x)$, $y \in Y$, представимы в виде смесей k_y компонент. Каждая компонента имеет n-мерную гауссовскую плотность с параметрами $\mu_{yj} = (\mu_{yj1}, \ldots, \mu_{yjn})$, $\Sigma_{yj} = \mathrm{diag}(\sigma^2_{yj1}, \ldots, \sigma^2_{yjn})$, $j = 1, \ldots, k_y$:

$$p_y(x) = \sum_{j=1}^{k_y} w_{yj} p_{yj}(x), \quad p_{yj}(x) = \mathcal{N}(x; \mu_{yj}, \Sigma_{yj}), \quad \sum_{j=1}^{k_y} w_{yj} = 1, \quad w_{yj} \geqslant 0.$$

Сети РБФ имеют ряд преимуществ перед рассмотренными многослойными сетями прямого распространения. Во-первых, они моделируют произвольную нелинейную функцию с помощью всего одного промежуточного слоя, тем самым избавляя разработчика от необходимости решать вопрос о числе слоев. Во-вторых, параметры линейной комбинации в вы-

ходном слое можно полностью оптимизировать с помощью хорошо известных методов линейной оптимизации, которые работают быстро и не испытывают трудностей с локальными минимумами, так мешающими при обучении с использованием алгоритма обратного распространения ошибки. Поэтому сеть РБФ обучается очень быстро.

```
rbf(x, ...) # Входят в пакет RSNNS
# S3 method for default
rbf(x, y, size = c(5), maxit = 100,
  initFunc = "RBF_Weights", initFuncParams = c(0, 1, 0, 0.02, 0.04),
  learnFunc = "RadialBasisLearning", learnFuncParams = c(1e-05, 0, 1e-05,
  0.1, 0.8), updateFunc = "Topological_Order", updateFuncParams = c(0),
  shufflePatterns = TRUE, linOut = TRUE, inputsTest = NULL,
  targetsTest = NULL, ...)

# NOT RUN {
demo(rbf_irisSnnsR)
# }
# NOT RUN {
demo(rbf_sin)
# }
# NOT RUN {
demo(rbf_sinSnnsR)
# }
# NOT RUN {

inputs <- as.matrix(seq(0,10,0.1))
outputs <- as.matrix(sin(inputs) + runif(inputs*0.2))
outputs <- normalizeData(outputs, "0_1")

model <- rbf(inputs, outputs, size=40, maxit=1000,
                initFuncParams=c(0, 1, 0, 0.01, 0.01),
                learnFuncParams=c(1e-8, 0, 1e-8, 0.1, 0.8), linOut=TRUE)

par(mfrow=c(2,1))
plotIterativeError(model)
plot(inputs, outputs)
lines(inputs, fitted(model), col="green")
# }

rbfDDA(x, ...)
## Default S3 method:
```

```
rbfDDA(x, y, maxit = 1, initFunc = "Randomize_Weights",
initFuncParams = c(-0.3, 0.3), learnFunc = "RBF-DDA",
learnFuncParams = c(0.4, 0.2, 5), updateFunc = "Topological_Order",
updateFuncParams = c(0), shufflePatterns = TRUE, linOut = FALSE, ...)

## Not run: demo(iris)
## Not run: demo(rbfDDA_spiralsSnnsR)
data(iris)
iris <- iris[sample(1:nrow(iris),length(1:nrow(iris))),1:ncol(iris)]
irisValues <- iris[,1:4]
irisTargets <- decodeClassLabels(iris[,5])
iris <- splitForTrainingAndTest(irisValues, irisTargets, ratio=0.15)
iris <- normTrainingAndTestSet(iris)
model <- rbfDDA(iris$inputsTrain, iris$targetsTrain)
summary(model)
plotIterativeError(model)
```

Задания для учащихся

Наименование рыбы	**Калории**	**Жиры**	**Холестерин**	**Натрий**	**Калий**	**Белок**	**Коэффициент скорости**
Рыба мечь	172	8	78	97	499	23	69
Тунец	184	6	49	50	323	30	68
Акула	130	4.5	51	79	160	21	70
лосось	208	13	55	59	363	20	58
скумбрия	262	18	75	83	401	24	59
треска	69	6.1	40	55	340	16	29
кефаль	88	2.2	53	72	468	23	28
пикша	90	0.6	66	261	351	20	37
палтус	186	14	46	80	268	14	55
камбала	70	1.9	45	296	160	12	44

Классификация рыб по скорости движения: 1. Очень быстроплавающие – коэффициент скорости от 61 до 70; 2. быстроплавающие– коэффициент скорости от 31 до 60; 3. Умеренно быстрые - – коэффициент скорости от 21 до 30

Классификация рыб по пищевой ценности: нежирные, умеренно жирные и жирные (характеристические показатели: калории, жиры, холестерин, натрий, калий, белок)

Вариант 1

Разработать классификатор для идентификации сортов рыбы с точки зрения пищевой ценности и скорости движения в воде, основанный на обучении без учителя. Визуализировать результаты с помощью карт Кохонена для оценки распределения сортов рыбы в рамках выбранного водного пространства. Полный список из 35 пунктов можно найти в приложении к лабораторной работе.
Использовать средства языка R – функции SOM и SOMGRID. Параметры для функции SOM GRID fct=”bubble”, topo=”rectangular”
Зададим число кластеров k=5 и выполним иерархическую кластеризацию (по умолчанию используются method = "complete" и distance = "euclidean"). Построить карты "mapping", "changes", "property", "quality".
Проверить результаты классификации с помощью функций RBF и rbfDDA.
Рыбы: тунец, лосось, пикша, палтус, треска, кефаль необходимо разделить по категориям: жирные и нежирные, очень быстроплавающие, быстроплавающие и умеренно быстрые.

Вариант 2

Разработать классификатор для идентификации сортов рыбы с точки зрения пищевой ценности и скорости движения в воде, основанный на обучении без учителя. Полный список из 35 пунктов можно найти в приложении к лабораторной работе.
Визуализировать результаты с помощью карт Кохонена для оценки распределения сортов рыбы в рамках выбранного водного пространства. Использовать средства языка R – функции SOM и SOMGRID. Параметры для функции SOM GRID fct=”gaussian”, topo=”hexagonal”
Зададим число кластеров k=6 и выполним иерархическую кластеризацию (по умолчанию используются method = "complete" и distance = "euclidean"). Построить карты "mapping", "quality", "property", "count".
Проверить результаты классификации с помощью функций RBF и rbfDDA.

Рыбы: акула, рыба - мечь, кефаль, треска, пикша, палтус, скумбрия необходимо разделить по категориям: умеренно жирные и нежирные, очень быстроплавающие, быстроплавающие и умеренно быстрые.

Вариант 3

Разработать классификатор для идентификации сортов рыбы с точки зрения пищевой ценности и скорости движения в воде, основанный на обучении без учителя. Визуализировать результаты с помощью карт Кохонена для оценки распределения сортов рыбы в рамках выбранного водного пространства. Полный список из 35 пунктов можно найти в приложении к лабораторной работе.

Использовать средства языка R – функции SOM и SOMGRID. Параметры для функции SOM GRID fct=”gaussian”, topo=”rectangular”
Зададим число кластеров k=7 и выполним иерархическую кластеризацию (по умолчанию используются method = "complete" и distance = "euclidean"). Построить карты "changes", "count", "dist.neighbours", "codes".
Проверить результаты классификации с помощью функций RBF и rbfDDA.

Рыбы: акула, тунец, кефаль, треска, пикша, палтус, камбала, лосось необходимо разделить по категориям : умеренно жирные и жирные, очень быстроплавающие, быстроплавающие и умеренно быстрые.

Вариант 4

Разработать классификатор для идентификации сортов рыбы с точки зрения пищевой ценности и скорости движения в воде, основанный на обучении без учителя. Визуализировать результаты с помощью карт Кохонена для оценки распределения сортов рыбы в рамках выбранного водного пространства. Полный список из 35 пунктов можно найти в приложении к лабораторной работе.

Использовать средства языка R – функции SOM и SOMGRID. Параметры для функции SOM GRID fct=”bubble”, topo=”hexagonal”
Зададим число кластеров k=9 и выполним иерархическую кластеризацию (по умолчанию используются method = "complete" и distance = "eu-

clidean"). Построить карты "changes", "count", "dist.neighbours", "property".
Проверить результаты классификации с помощью функций RBF и rbfDDA.

Рыбы: акула, рыба - мечь, кефаль, треска, пикша, палтус, камбала, скумбрия необходимо разделить по категориям : умеренно жирные и жирные, очень быстроплавающие, быстроплавающие и умеренно быстрые.

Вариант 5

Разработать классификатор для идентификации сортов рыбы с точки зрения пищевой ценности и скорости движения в воде, основанный на обучении без учителя. Визуализировать результаты с помощью карт Кохонена для оценки распределения сортов рыбы в рамках выбранного водного пространства. Полный список из 35 пунктов можно найти в приложении к лабораторной работе.
Использовать средства языка R – функции SOM и SOMGRID. Параметры для функции SOM GRID fct=”bubble”, topo=”hexagonal”
Зададим число кластеров k=7 и выполним иерархическую кластеризацию (по умолчанию используются method = "complete" и distance = "euclidean"). Построить карты " dist.neighbours ", "changes", "codes", "quality".
Проверить результаты классификации с помощью функций RBF и rbfDDA.

Рыбы: тунец, лосось, пикша, палтус, треска, кефаль необходимо разделить по категориям : жирные и умеренно жирные, очень быстроплавающие, быстроплавающие и умеренно быстрые.

Вариант 6

Необходимо составить карты предпочтений при выборе услуг обслуживания автомобилей в трех сервисных автоцентрах «Дмитровка», «На Калужской», «Мичуринский».

Вид услуги	«Дмитровка», время выполнения	«На Калужской», время	«Мичуринский», время	«Дмитровка», цена	«На Калужской», цена	«Мичуринский», цена
Замена сальника коленчатого вала	12	11.8	12.4	19200	19400	21000
Замена клапанной крышки	4.6	5.2	4.2	7360	6900	7700
Замер компрессорного давления	2	2.1	1.89	3200	3400	3000
Диагностика двигателя	1.14	1.27	0.89	1600	1630	1737
Диагностика ходовой части	0.87	1.2	1.23	1630	1609	1670
Проверка угла установки колес	0.55	0.48	0.69	860	790	900
Замена амортизаторов	3	3.5	3.2	6200	6400	6100
Замена масла в резервуаре заднего моста	1.5	1.3	1.4	2400	2500	2600

Замена сальника хвостовика редуктора заднего моста	2.5	2.6	2.8	4000	4200	4150
Ремонт коробки передач	5	4.9	5.2	8000	7800	7940

Анализировать виды услуг: с 1 по 5 из таблицы. Выбрать лучший автосервис с точки зрения цены на выполненные работы. Полный список из 35 пунктов можно найти в приложении к лабораторной работе.
Визуализировать результаты с помощью карт Кохонена.
Использовать средства языка R – функции SOM и SOMGRID. Параметры для функции SOM GRID fct=”bubble”, topo=”hexagonal”
Проверить результаты классификации с помощью функций RBF и rbfDDA.

Зададим число кластеров k=9 и выполним иерархическую кластеризацию (по умолчанию используются method = "complete" и distance = "euclidean"). Построить карты "changes", "count", "dist.neighbours", "property".

Вариант 7

Необходимо составить карты предпочтений при выборе услуг обслуживания автомобилей в трех сервисных автоцентрах «Дмитровка», «На Калужской», «Мичуринский».
Анализировать виды услуг: с 6 по 10 из таблицы. Выбрать лучший автосервис с точки зрения скорости выполнения работ. Полный список из 35 пунктов можно найти в приложении к лабораторной работе.

Визуализировать результаты с помощью карт Кохонена.
Использовать средства языка R – функции SOM и SOMGRID. Параметры для функции SOM GRID fct=”gaussian”, topo=”hexagonal”
Зададим число кластеров k=6 и выполним иерархическую кластеризацию (по умолчанию используются method = "complete" и distance = "euclidean"). Построить карты "mapping", "quality", "property", "count".

Проверить результаты классификации с помощью функций RBF и rbfDDA.

Вариант 8

Необходимо составить карты предпочтений при выборе услуг обслуживания автомобилей в трех сервисных автоцентрах «Дмитровка», «На Калужской», «Мичуринский».
Анализировать виды услуг: с 3 по 7 из таблицы. Выбрать лучший автосервис с точки зрения ценовых затрат на выполнение работ. Полный список из 35 пунктов можно найти в приложении к лабораторной работе.

Визуализировать результаты с помощью карт Кохонена.
Использовать средства языка R – функции SOM и SOMGRID. Параметры для функции SOM GRID fct=”bubble”, topo=”rectangular”
Зададим число кластеров k=5 и выполним иерархическую кластеризацию (по умолчанию используются method = "complete" и distance = "euclidean"). Построить карты "mapping", "changes", "property", "quality".
Проверить результаты классификации с помощью функций RBF и rbfDDA.

\.
Вариант 9

Необходимо составить карты предпочтений при выборе услуг обслуживания автомобилей в трех сервисных автоцентрах «Дмитровка», «На Калужской», «Мичуринский».
Анализировать виды услуг: с 2, 4, 6, 8, 10 из таблицы. Выбрать лучший автосервис с точки зрения двух показателей скорость и цена на выполнение работ. Полный список из 35 пунктов можно найти в приложении к лабораторной работе.
Визуализировать результаты с помощью карт Кохонена.
Использовать средства языка R – функции SOM и SOMGRID. Параметры для функции SOM GRID fct=”bubble”, topo=”hexagonal”
Зададим число кластеров k=10 и выполним иерархическую кластеризацию (по умолчанию используются method = "complete" и distance = "euclidean"). Построить карты "changes", "count", "dist.neighbours", "property".

Проверить результаты классификации с помощью функций RBF и rbfDDA.

Вариант 10

Наименование неисправности	Компрессия Двигателя с открытой заслонкой, мПА	Компрессия Двигателя с закрытой заслонкой, мПА	Примечание
Двигатель исправен	1 -1.2	0.6 -0.8	
Трещина в перемычке поршня	0.6 – 0.8	03. – 0.4	
Прогар поршня	0.5	0.1	
Залегание колец в канавках поршня	0.2 – 0.4	0.2	
Задир поршня	0.2 – 0.8	0.15 – 0.5	
Задир цилиндра	0.3 -0.8	0.4 -0.5	
Деформация клапана	0.3 – 0.7	0.2	
Прогар клапана	0.1 -0.4	0.01	
Дефект профиля кулачка распредвала	0.7 – 0.8	0.1 – 0.3	
Сильный нагар в камере сгорания	1.2 – 1.5	0.9 – 1.2	
Износ деталей поршневой группы	0.6 - 0.9	0.4 – 0.6	
Зависание клапана	0.4 – 0.8	0.2 - 0.4	

Необходимо идентифицировать неисправности двигателя автомобиля (задир поршня, Двигатель исправен, Зависание клапана, прогар поршня, деформация клапана, задир цилиндра) по двум показателям. Полный список из 40 пунктов можно найти в приложении к лабораторной работе.
Визуализировать результаты с помощью карт Кохонена.
Зададим число кластеров k=12 и выполним иерархическую кластеризацию (по умолчанию используются method = "complete" и distance = "euclidean"). Построить карты " dist.neighbours ", "changes", "codes", "quality".

Проверить результаты классификации с помощью функций RBF и rbfDDA.

Вариант 11

Необходимо идентифицировать неисправности двигателя автомобиля (Трещина в перемычке поршня, Двигатель исправен, Зависание клапана, прогар поршня, износ деталей поршневой группы, сильный нагар в камере сгорания, дефект профиля кулачка распредвала)по двум показателям. Полный список из 37 пунктов можно найти в приложении к лабораторной работе.
Визуализировать результаты с помощью карт Кохонена.
Использовать средства языка R – функции SOM и SOMGRID. Параметры для функции SOM GRID fct=”bubble”, topo=”rectangular”
Зададим число кластеров k=4 и выполним иерархическую кластеризацию (по умолчанию используются method = "complete" и distance = "euclidean"). Построить карты "mapping", "changes", "property", "quality".
Проверить результаты классификации с помощью функций RBF и rbfDDA.

Вариант 12

Наименование показателя низового пожара	Слабый низовой	Средний низовой	Сильный низовой

Скорость распространения огня	До 1	1-3	Более 3
Высота пламени	До 0.5 м	0.5 – 1.5	Более 1.5

Наименование показателя верхового пожара	Слабый верховой	Средний верховой	Сильный верховой
Скорость распространения огня	До 3	3 - 100	Более 100
Высота пламени	До 2 м	2 – 4	Более 4 м

Наименование показателя подземного пожара	Слабый подземный	Средний подземный	Сильный подземный
Скорость распространения огня	До 0.65	0.65 - 2	Более 2
Высота пламени	До 0.55 м	0.55 – 1.2	Более 1.2 м

Необходимо идентифицировать вид низового пожара на выбранной лесистой местности (слабый, средний, сильный) по двум показателям: скорость огня и высота пламени. Полный список из 38 пунктов можно найти в приложении к лабораторной работе.

Визуализировать результаты с помощью карт Кохонена.

Использовать средства языка R – функции SOM и SOMGRID. Параметры для функции SOM GRID fct=”bubble”, topo=”hexagonal”

Зададим число кластеров k=7 и выполним иерархическую кластеризацию (по умолчанию используются method = "complete" и distance = "euclidean"). Построить карты " dist.neighbours ", "changes", "codes", "quality".

Проверить результаты классификации с помощью функций RBF и rbfDDA.

Вариант 13

Необходимо идентифицировать вид верхового пожара на выбранной лесистой местности (слабый, средний, сильный) по двум показателям: скорость огня и высота пламени. Полный список из 42 пунктов можно найти в приложении к лабораторной работе.
Визуализировать результаты с помощью карт Кохонена.
Использовать средства языка R – функции SOM и SOMGRID. Параметры для функции SOM GRID fct=”gaussian”, topo=”hexagonal”
Зададим число кластеров k=16 и выполним иерархическую кластеризацию (по умолчанию используются method = "complete" и distance = "euclidean"). Построить карты "mapping", "quality", "property", "count".

Проверить результаты классификации с помощью функций RBF и rbfDDA.

Вариант 14

Необходимо идентифицировать вид подземного пожара на выбранной лесистой местности (слабый, средний, сильный) по двум показателям: скорость огня и высота пламени. Полный список из 33 пунктов можно найти в приложении к лабораторной работе.
Визуализировать результаты с помощью карт Кохонена.
Использовать средства языка R – функции SOM и SOMGRID. Параметры для функции SOM GRID fct=”gaussian”, topo=”rectangular”
Зададим число кластеров k=10 и выполним иерархическую кластеризацию (по умолчанию используются method = "complete" и distance = "euclidean"). Построить карты "changes", "mapping", "dist.neighbours", "codes".
Проверить результаты классификации с помощью функций RBF и rbfDDA.

Вариант 15

Характеристика ветра	Значение показателя	Горизонтальный Барический градиент атмосферного давления	Вертикальный Барический градиент атмосферного давления
Слабый	0.5	1.1	1.24
Умеренный	6-14	1.65	1.54
Сильный	15-24	2.33	2.79
Очень сильный	25-32	2.64	2.89
Ураганный	Более 33	2.92	2.97

Показатель облачности	Значение
Малая	До 3 баллов
Переменная	4-7
С прояснениями	7-8
Облачно	8-10

Показатель насыщенности (плотности) дождя	Значение	Показатель продолжительности дождя	Значение
Отсутствует	Менее 0.2	Кратковременный	Менее 3 ч
Моросящий	0.3 – 10	Временный	От 3 до 6 ч
Сильный	11 – 49	Продолжительный	Более 6 ч

Необходимо разработать кратковременный прогноз погоды – наличие или отсутствие дождя на пять дней (дождь может быть временным и сильным, моросящим и продолжительным и т. Д.). Показатели прогноза погоды: облачность, сила ветра, барические градиенты горизонтальный и вертикальный. Полный список из 44 пунктов можно найти в приложении к лабораторной работе.

Визуализировать результаты с помощью карт Кохонена.

Использовать средства языка R – функции SOM и SOMGRID. Параметры для функции SOM GRID fct=”bubble”, topo=” rectangular ”

Зададим число кластеров k=9 и выполним иерархическую кластеризацию (по умолчанию используются method = "complete" и distance = "euclidean"). Построить карты "quality ", "count", "dist.neighbours", "property".

Проверить результаты классификации с помощью функций RBF и rbfDDA.