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 № 1
по курсу «Методы машинного обучения»

ЛИНЕЙНЫЕ КЛАССИФИКАТОРЫ

Калуга
2018

Теоретические основы.

Язык программирования R является диалектом (реализацией) языка S, который был разработан как эффективное и удобное средство для работы с данными, моделирования и визуализации. R развивается в рамках opensource-проекта и доступен для различных платформ: Windows, MacOS и UNIX/UNIX-подобных систем (включая FreeBSD и Linux).

Основным способом работы пользователя в системе R является использование интерактивного режима работы в консоли интерпретатора, либо написание программ - скриптов для их дальнейшего исполнения в системе. Как правило, скрипты сохраняются в файлах, имеющих расширение .R.

Каждая команда или выражение в языке R возвращает некоторый результат. Выражение может быть частью другого выражения. В частности, возможны множественные присваивания. Выражения можно объединять в блоки. Блок – это несколько выражений, заключенных в фигурные скобки. Сами выражения, как обычно, разделяются точкой с запятой или символом перехода на новую строку. Результатом выполнения всего блока будет результат последнего из выражений, составляющих блок. Для получения справки по той или иной функции можно выполнить команду help("function_name").

Векторы являются основополагающей структурой данных языка R, представляющей собой некоторое количество однотипных элементов непрерывно расположенных в памяти. В R есть 6 базовых типов векторов (или, что тоже самое, их элементов): логический (logical), целочисленный (integer), вещественный (double), комплексный (complex), символьный (character) и двоичные данные (raw). Любая константа и переменная одного из перечиленных выше типов также является вектором длины 1. Для определения длины вектора служит функция length(x).

В то время как векторы представляют одномерную последовательность данных базовых типов, зачастую удобнее оперировать многомерными данными. Так матрицы в языке R представляют собой двумерные векторы. Для преобразования вектора в матрицу можно присвоить вектору атрибут размерности с помощью функции dim(x). Данная функция принимает в качестве аргумента вектор, матрицу или массив, возвращая вектор размерностей. Вектор может иметь либо пустой вектор размерностей, либо вектор размерностей длины один, содержа в себе длину вектора.

Дальнейшим развитием идеи многомерности данных в языке R являются массивы, которые могут иметь произвольное количество размерностей. Работа с массивами почти идентична работе с матрицами.

Как уже отмечалось выше, список является общим (generic) вектором, т.е. его элеметы могут иметь различный тип. Элементами списка могут быть в том числе векторы и другие списки. Для создания списка служит функция list(...), принимающая на вход произвольное количество объектов, которые требуется объединить в список.

Факторы представляют собой структуру данных для хранения векторов категориальных данных (классов), т.е. величин, которые могут принимать значения из конечно, в общем случае, неупорядоченного множества. Факторы создаются с помощью функции factor(x = character(), levels, labels = levels).

Фреймы данных (data frames) один из самых важных типов данных в R, позволяющий объединять данные разных типов вместе. Фрейм является специальной версией списка, где все элементы имеют одинаковую длину. Можно считать, что фрейм данных это двумерная таблица, в которой (в отличие от числовых матриц), разные столбцы могут содержать данные разных типов (но все данные в одном столбце имеют один тип). Например, такая таблица может содержать результаты эксперимента.

Создать фрейм данных можно с помощью функции data.frame(...), аргументами которой являются произвольное количество элементов (столбцов) фрейма. В качестве элементов фрейма данных могут выступать векторы, факторы, матрицы, списки или другие фреймы. При этом все векторы должны иметь одинаковую длину, а матрицы и фреймы одинаковое (такое же) число строк. Функция data.frame(...) просто собирает все данные вместе.

Функции в языке R определяются с помощью ключевого слова function, вслед за которым в круглых скобках идет перечисление формальных аргументов функции. Аргументам можно назначать значения по умолчанию, как выражения, зависящие, в том числе, и от других формальных аргументов. При этом вычисление данного выражения будет производиться только, если значение этого аргумента потребуется при выполнении тела функции. Тело функции идет после указания списка ее формальных аргументов и должно быть заключено в фигурные скобки если содержит более одного выражения. Значением, возвращаемым функцией является либо результат последнего выражения в теле функции, либо аргумент функции return(x), прерывающей выполнение функции.

Зачастую в ходе статистического моделирования необходимо выразить зависимость одной величины от другой и описать характер данной зависимости. В языке R для это используются специальные объекты, на-

зываемые формулами. Для определения формулы используется оператор ~, синтаксис которого выглядит следующим образом: y ~model. Слева от оператора ~ указывается имя зависимой переменной, а справа выражение специального вида – модель зависимости. Например, формула y ~x1 + x2 определяет линейную зависимость y от x1 и x2: $y = a_0 + a_1x_1 + a_2x_2$. Все переменные, используемые в модели, должны быть либо числовыми векторами, либо факторами. При этом логические векторы преобразуются в факторы.

Для изображения графиков в R есть функция plot(x, y, ...). Здесь x – вектор значений абсцисс и y – вектор значений ординат. Функция plot допускает множество дополнительных параметров. Приведем описание некоторых из них:

- log = "x" или log = "y" или log = "xy" делает соответственно масштаб по оси абсцисс, ординат или по обеим осям логарифмическим;
- type позволяет указать тип выводимого графика. Некоторые значения параметра: "p" (точки, по умолчанию), "l" (линии), "b" (линии и точки), "o" (линии и точки перекрываются), "n" (ничего не рисуется);
- xlab и ylab позволяет указать подписи к оси абсцис и ординат соответственно;
- main и sub создают надписи сверху и снизу графика;
- col задает цвет графика. Некотороые возможные значения: "blue", "red", "green", "cyan", "magenta", "yellow", "black". Можно задать цвет rgbвектором функцией rgb(r, g, b), где r, g и b от 0 до 1.

Машина опорных векторов (SVM, Support Vector Machine) является одним из наиболее популярных и универсальных алгоритмов машинного обучения. Данный алгоритм может применяться как для решения задач классификации, так и для восстановления регрессии. Решение задачи бинарной классификации $y \in \{-1,1\}$ методом опорных векторов сводится к отысканию гиперплоскости $\beta^T h(x) + \beta_0 = 0$ в спрямляющем пространстве, которая оптимальным образом разделяет точки из обучающей выборки разных классов. Коэффициенты гиперплоскскти выбираются как результат решения следующей оптимизационной задачи:

$$\hat{\beta}, \hat{\beta}_0 = \arg\min_{\beta,\beta_0} C \sum_{i=1}^{n} w_{y_i}[1 - y^{(i)}(\beta^T h(x^{(i)}) + \beta_0)]_+ + \frac{1}{2}\|\beta\|_2^2,$$

где C — неотрицательный параметр алгоритма обучения и

$$[z]_+ = \begin{cases} z, & \text{если } z > 0; \\ 0, & \text{иначе.} \end{cases}$$

Решение данной задачи представляется в следующем виде:

$$\hat{\beta} = \sum_{i=1}^{n} \alpha_i y^{(i)} h(x^{(i)}).$$

А решающее правило принимает вид

$$f(x) = \text{sign}(\sum_{i=1}^{n} \alpha_i y^{(i)} K(x^{(i)}, x) + \hat{\beta}_0)$$

где $\alpha_i \geq 0$ и сторого больше нуля для опорных векторов, $K(x, x') = \langle h(x), h(x') \rangle$ — скалярное произведение соответствующих точек в спрямляющем пространстве (ядро).

Для задачи восстановления регрессии машина опорных векторов строит функцию вида $f(x) = \beta^T h(x) + \beta_0$, где коэффициенты являются решением оптимизационной задачи

$$\hat{\beta}, \hat{\beta}_0 = \arg\min_{\beta,\beta_0} C \sum_{i=1}^{n} V_\varepsilon(y^{(i)} - \beta^T h(x^{(i)}) - \beta_0) + \frac{1}{2}\|\beta\|_2^2, \tag{5}$$

где

$$V_\varepsilon(z) = \begin{cases} 0, & \text{если } |z| < \varepsilon; \\ |z| - \varepsilon, & \text{иначе.} \end{cases} \tag{6}$$

Конечное решающее правило имеет вид

$$f(x) = \sum_{i=1}^{n} \alpha_i K(x^{(i)}, x) + \hat{\beta}_0, \tag{7}$$

где $\alpha_i \geq 0$ и сторого больше нуля для опорных векторов.

Реализация для языка R алгоритма обучения машины опорных векторов доступна в пакете e1071. Рассмотрим основы работы с данным пакетом.

Для обучения SVM-модели предназначена функция svm имеющая два варианта интерфейса:

```
svm(formula, data = NULL, ..., subset, na.action = na.omit,
    scale = TRUE)

svm(x, y = NULL, scale = TRUE, type = NULL, kernel = "radial",
    degree = 3, gamma = if (is.vector(x)) 1 else 1 / ncol(x),
    coef0 = 0, cost = 1, nu = 0.5, class.weights = NULL,
    cachesize = 40, tolerance = 0.001, epsilon = 0.1, shrinking
    = TRUE, cross = 0, probability = FALSE, fitted = TRUE, seed
    = 1L, ..., subset, na.action = na.omit)
```

Параметрами данных функций являются:

- formula — формула, описывающая восстанавливаемую зависимость;
- data — фрейм данных или список, содержащий переменные, использованные в символическом описании модели formula. Если data = NULL имена, использованные в formula, должны быть доступны в текущем рабочем пространстве;
- x — матрица, вектор или разреженная матрица (объект класса matrix.csr, реализованного к пакете SparseM), содержащая признаковые описания объектов обучающей выборки;
- y — фактор (для задачи классификации) или числовой вектор (для восстановления регрессии) со значениями целевого признака для каждой строки/компоненты x. В случае, если количество целевых классов больше двух, то решается серия задач бинарной классификации по схеме «каждый против каждого», и конечное решение принимается голосованием;
- scale — логический вектор, определяющий для каких признаков следует выполнять масштабирование перед обучением машины опорных векторов. Если длина вектора scale равна единице, то указанное значение распространяется на все признаки. По умолчанию, данные (как x, так и y) масштабируются таким образом, чтобы получить математическое ожидание, равное нулю, и дисперсию, равную единице;

- type — тип оптимизационной задачи, решаемой при обучении SVM. Рассмотренная выше постановка задачи классификации соответствует типу "C-classification", восстановления регрессии — типу "eps-regression". Также доступны типы "nu-classification", "nu-regression"
-
- **kernel** — тип используемого ядра. Допустимы следующие значения:
  "linear" соответсвует ядру $K(u,v) = \langle u, v\rangle$,
  "polynomial" — $K(u,v) = (gamma\langle u, v\rangle + coef0)^{degree}$,
  "radial" — $K(u,v) = \exp(-gamma\|u - v\|_2^2)$,
  "sigmoid" — $K(u,v) = \text{th}(gamma\langle u, v\rangle + coef0)$;
- degree, gamma, coef0 — параметры функции ядра;
- cost — параметр C (см. (1) и (5));
- epsilon — параметр функции потерь алгоритма ε-регрессии;
- class.weights — поименованный вектор весов для различных классов (w в (1)). Веса, не указанные в class.weights, считаются равными единице;
- nu — параметр, используемый при значении type, равном "nu-classification",
  "nu-regression" или "one-classification";
- cachesize — размер памяти (в мегабайтах), используемой для кэширования вычислений;
- tolerance — пороговое значение, задающее критерий остановки по точности для итерационного метода оптимизации;
- shrinking — логическое значение, определяющее будет ли использована эвристика для уменьшения размерности решаемой задачи оптимизации;
- cross — если положительное целое число, то для оценки качества модели будет выполнен cross-кратный перекрестный контроль. При этом показателем качества решения задачи классификации является доля правильно классифицированных прецедентов, для регрессии — среднеквадратическая

ошибка. Если cross = 0, то перекрестный контроль не будет выполнен;

- probability — логическое значение, определяющее следует ли на этапе обучения вычислять значения, которые в дальнейшем позволят оценить веротности принадлежности нового объекта каждому из рассматриваемых классов (в случае классификации) и восстановить распределение ошибки предсказания в заданной точке (в случае восстановления регрессии);

- fitted — логическое значение, которое определяет, требуется ли включать предсказанные значения целевого признака для объектов обучающей выборки в результирующую модель;

- seed — целое число, значение для инициализации генератора псевдослучайных чисел, который используется при осуществлении перекрестного контроля и оценке параметров вероятностных распределений, которые требуют 5-кратного перекрестного контроля;

- subset — вектор индексов прецедентов, которые необходимо использовать на этапе обучения;

- na.action — функция, определяющая действие, которое должно быть выполнено при обнаружении отсутствующих значений. По умолчанию (na.action = na.omit) прецеденты с отсутствующими значениями используемых признаков игнорируются, т.е. не используются для обучения.

Функция svm возвращает список (объект класса svm), который содержит следующие элементы:

- call — строка вызова функции svm;

- type, cost, epsilon, nu, degree, gamma, coef0, na.action — значения соответсвующих параметров, использовавшихся для обучения;

- scaled — логический вектор, определяющий какие признаки были масштабированы перед обучением;

- x.scale, y.scale — параметры масштабирования признаков из x и y;

- nclasses, levels, labels — количество целевых классов и их обозначения;
- tot.nSV — общее количество опорных векторов;
- nSV — количество опорных векторов, принадлежащих каждому из рассматриваемых целевых классов;
- SV — опорные векторы (возможно масштабированные);
- index — индексы опорных векторов в обучающей выборке;
- `decision.values` — величины $\hat{\beta}^T h(x^{(i)}) + \hat{\beta}_0$. Для задачи классификации данные значения вычисляются для каждой задачи бинарной классификации;
- fitted — вектор предсказаний модели для объектов обучающей выборки;
- residuals — разности между предсказаниями модели для объектов обучающей выборки и истинными значениями целевого признака;
- coefs — оцененные коэффициенты β для задачи восстановления регрессии и коэффициенты β, умноженные на соответствующие значения γ ;
- rho — оцененное значение параметра $-\beta_0$;
- compprob, probA, probB, sigma — значения, необходимые для оценки распределений целевого признака для заданного объекта x;
- sparse — логическое значение, определяющее была ли обучающая выборка представлена в разреженном формате.

Для осуществления предсказаний на новых данных с помощью обученной SVM-модели в пакете e1071 служит функция predict:

```
predict(object, newdata, decision.values = FALSE, probability =
    FALSE, ..., na.action = na.omit)
```

Данная функция принимает следующие аргументы:

- `object` — объект класса `svm`, обученная SVM-модель;
- `newdata` — данные, на которых требуется выполнить предсказания;
- `decision.values` — логическое значение, определяющее следует ли возвращать наряду с предсказаниями модели величины $\hat{\beta}^T h(x^{(i)}) + \hat{\beta}_0$;
- `probability` — логическое значение, определяющее следует ли наряду с прдесказаниями модели вычислять вероятности принадлжености рассматриваемого объекта к каждому из целевых классов;
- `na.action` — функция, определяющая действие, которое должно быть выполнено при обнаружении отсутствующих значений.

Для наглядного представления SVM-модели пакет e1071 предоставляет функцию plot для отображения разбиения исходного пространства признаков на области:

```
plot(x, data, formula, fill = TRUE, grid = 50, slice = list(),
    symbolPalette = palette(), svSymbol = "x", dataSymbol =
    "o", ...)
```

Данная функция принимает следующие параметры:

- `x` — объект класса `svm`;
- `data` — обучающая выборка;

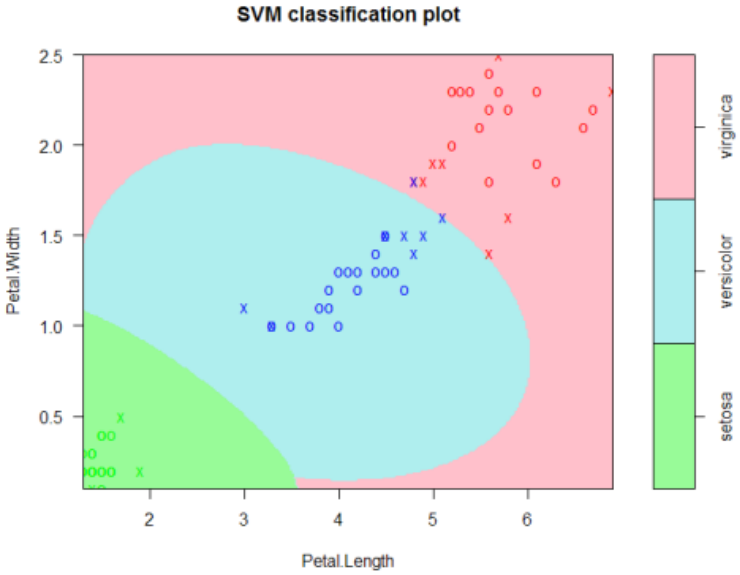
- `formula` — формула, определяющая два признака для визуализации. Если общее количество признаков равно двум, то данный параметр не обязателен;
- `fill` — логическое значение, определяющее следует ли раскрашивать пространство признаков в соответствии с предсказываемым значением SVM-классификатора. Если `fill = TRUE`, то для каждой точки сетки размера `grid` × `grid`, будет выполнена классификация и в зависимости от результата точка будет окрашена в некоторый цвет;
- `grid` — разрешение сетки для классификации с целью раскрашивания пространства признаков;
- `slice` — именнованный список, который определяет константные значения невизуализируемых признаков;
- `symbolPalette` — палитра, определяющая раскраску точек обучающей выборки;
- `svSymbol` — символ, соответствующий опорным векторам;
- `dataSymbol` — символ, соответствующий объектам обучающей выборки, не являющимися опорными векторами;
- `...` — другие параметры, которые будут переданы в функции `filled.contour` и `plot`.

Рассмотрим задачу классификации сортов ирисов по параметрам цветков (набор данных iris). Набор данных содержит 4 предикативных признака Sepal.Length, Sepal.Width, Petal.Length, Petal.Width и один целевой Species. Всего рассматривается 3 целевых класса. Попытаемся восстановить зависимость признака Species от Petal.Length и Petal.Width, не принимая во внимания остальные признаки. Случайным образом разобьем выборку на две части: обучающую и тестовую. Последовательно обучим SVM-модели с линейным и радиальным ядрами, нарисуем разбиения пространства признаков с помощью полученных моделей, подсчитаем количество ошибок классификации на обучающей и тестовой выборках.

```
> library(e1071)
> area.pallete = function(n = 3)
+ {
+      cols = rainbow(n)
+      cols[1:3] = c("PaleGreen", "PaleTurquoise", "Pink")
+      return(cols)
+ }
> symbols.pallete = c("Green", "Blue", "Red")
>
> dataIris = iris[c("Petal.Width", "Petal.Length", "Species")]
> plot(Petal.Width ~ Petal.Length, dataIris, col = Species)
>
> set.seed(0)
```

```
> trainIdx = sample(nrow(dataIris), nrow(dataIris) / 2, replace
    = FALSE)
> dataIrisTrain = dataIris[trainIdx, ]
> dataIrisTrainObjects = dataIris[trainIdx, c("Petal.Width",
    "Petal.Length")]
> dataIrisTestObjects = dataIris[-trainIdx, c("Petal.Width",
    "Petal.Length")]
>
> svmModelLinear = svm(Species ~ ., data = dataIrisTrain, type
    = "C-classification", cost = 1, kernel = "linear")
> plot(svmModelLinear, dataIrisTrain, grid = 250, symbolPalette
    = symbols.pallete, color.palette = area.pallete)
> predictionsTrain = predict(svmModelLinear,
    dataIrisTrainObjects)
> table(dataIrisTrain$"Species", predictionsTrain)
             predictionsTrain
              setosa versicolor virginica
  setosa          25          0         0
  versicolor       0         25         1
  virginica        0          1        23
> predictionsTest = predict(svmModelLinear, dataIrisTestObjects)
> table(dataIrisTest$"Species", predictionsTest)
             predictionsTest
              setosa versicolor virginica
  setosa          24          0         0
  versicolor       1         18         1
  virginica        0          8        23
>
> svmModelRBF = svm(Species ~ ., data = dataIrisTrain, type =
    "C-classification", cost = 1, kernel = "radial", gamma = 1)
> plot(svmModelRBF, dataIrisTrain, grid = 250, symbolPalette =
    symbols.pallete, color.palette = area.pallete)
> predictionsTrain = predict(svmModelLinear,
    dataIrisTrainObjects)
> table(dataIrisTrain$"Species", predictionsTrain)
             predictionsTrain
```

```
set.seed(0)
x = seq(0.1, 5, by = 0.05)
y = log(x) + rnorm(x, sd = 0.3)
plot(x, y)

svmModel = svm(x, y, type = "eps-regression", eps = 0.25, cost
    = 1)
points(x[svmModel$index], y[svmModel$index], col = "red")
predctions = predict(svmModel, x)
lines(x, predctions, col = "dodgerblue", lwd = 2)
lines(x, predctions + svmModel$epsilon, col = "cyan")
lines(x, predctions - svmModel$epsilon, col = "cyan")
```



SVM в задачах классификации

Пусть обучающая выборка хранится в файле svmdata.txt, тестовая выборка — к файле svmdatatest.txt.

Загрузим пакет, содержащий функцию svm:

> library(e1071)

Загрузим обучающую выборку в data.frame SVMData из файла:

> SVMData < - read.table("svmdata.txt")

Вызовем функцию обучения svm с ядром linear и значением константы C, равным 2: model <- svm(SVMData$Color ~., SVMData, kernel="linear", cost=2) Построим график, иллюстрирующий результаты обучения (крестиками обозначены опорные вектора):

> plot(model, SVMData)

Найдем число ошибок на обучающей выборке. Для предсказания с использованием обученной модели используется функция predict, для отображения количества ошибок предсказания — функция table.

> x <- subset(SVMData, select = -Color)
> Color_pred <- predict(model, x)
> table(SVMData$Color, Color_pred)

Оценим качество обучения по количеству ошибок на тестовой выборке.

> SVMDataTest <- read.table("svmdatatest.txt")
> x <- subset(SVMDataTest, select = -Color)
> Color_pred <- predict(model, x)
> table(SVMDataTest$Color, Color_pred)

1.1.5. SVM в задаче восстановления регрессии

Сгенерируем случайную обучающую выборку:

```
> x < - seq(0.1, 5, by = 0.05)
> y < - log(x) + rnorm(x, sd = 0.2)
```

Вызовем функцию обучения `svm` в режиме `eps-regression` и значением константы ε, равным `0.15`:

```
> model < - svm(x, y, type = "eps-regression", eps = 0.15)
```

Оценим качество обучения на обучающей выборке:

```
> y_pred < - predict(model, x)
> plot(x, y)
> points(x, log(x), col = 2)
> points(x, y_pred, col = 4)
```

Метод опорных векторов пользуется огромной популярностью среди исследователей и разработчиков программного обеспечения. Создано огромное количество практических реализаций данного метода.

Так функциональную форму SVM можно представить в виде — **f(x,y)=ax+by+c**. Затем вводится новый прогрессивный теоретический термин оценки метода опорных векторов «**Характеристика силы (мощности)**».

Предположим, что настройка произвольного параметра, выглядит так, a = 1, b = -2, c = -1. Тогда, если мы подаем точку [1.2, 0.7],SVM вычислит значение 1 * 1.2 + (-2) * 0.7 — 1 = -1.2. Эта точка помечена как +1 в обучающих данных, поэтому мы хотим, чтобы значение было больше чем 1. Градиент в верхней части схемы будет в таком случае положительным: +1, и будет выполнять обратное распространение ошибки к a,b,c.

Кроме того, будет иметь место регуляризационное натяжение на a с силой -1 (чтобы сделать его меньше) и регуляризационное натяжение на b с силой +2, чтобы сделать его больше, в направлении нуля.

Вместо этого предположим, что подается на SVM точка ввода данных [-0.3, 0.5]. Она вычисляет 1 * (-0.3) + (-2) * 0.5 — 1 = -2.3. Метка для этой точки имеет значение -1, а так как -2.3 меньше, чем -1, мы понимаем, что в соответствии с нашей характеристикой силы SVM должна радоваться: вычисленное значение будет крайне отрицательным и будет соответствовать отрицательной метке в этом примере. В конце схемы не будет натяжения (т.е. оно будет нулевым), так как изменения не нужны. Однако, все равно будет присутствовать регуляризационное натяжение на a с силой -1 и на b с силой +2.

Рассмотрим пример реализации алгоритма **спуска стохастического градиента:**

```
var data = []; var labels = [];
data.push([1.2, 0.7]); labels.push(1);
data.push([-0.3, -0.5]); labels.push(-1);
data.push([3.0, 0.1]); labels.push(1);
data.push([-0.1, -1.0]); labels.push(-1);
data.push([-1.0, 1.1]); labels.push(-1);
data.push([2.1, -3]); labels.push(1);
var svm = new SVM();

// функция, которая рассчитывает точность классификации
var evalTrainingAccuracy = function() {
  var num_correct = 0;
  for(var i = 0; i < data.length; i++) {
    var x = new Unit(data[i][0], 0.0);
    var y = new Unit(data[i][1], 0.0);
    var true_label = labels[i];

    // смотрим, совпадает ли прогноз с имеющейся меткой
    var predicted_label = svm.forward(x, y).value > 0 ? 1 : -1;
    if(predicted_label === true_label) {
      num_correct++;
    }
  }
  return num_correct / data.length;
};

// петля обучения
```

```
for(var iter = 0; iter < 400; iter++) {
  // подбираем произвольную точку ввода данных
  var i = Math.floor(Math.random() * data.length);
  var x = new Unit(data[i][0], 0.0);
  var y = new Unit(data[i][1], 0.0);
  var label = labels[i];
  svm.learnFrom(x, y, label);

  if(iter % 25 == 0) { // каждые 10 повторений...
    console.log('training accuracy at iter ' + iter + ': ' + evalTrainingAccuracy());
  }
}
```

Теперь рассмотрим пример из области финансовых рынков. Для анализа возьмем индекс S&P500, его значения могут быть загружены автоматически в программу на R с помощью модуля quantmod. А реализацию метода опорных векторов найдем в пакете e1071.

Сначала надо провести подготовительную работу с данными, преобразовать их в удобную для формализации форму:

```
library(quantmod)
library(e1071)

#Будем исследовать индекс S&P500
getSymbols("^GSPC", from='2005-01-01')
data <- GSPC

#Выберем побольше разных индикаторов
atr <- ATR(data)[,'atr']
ema7 <- EMA(Cl(data), n=7)[,1]
ema14 <- EMA(Cl(data), n=14)[,1]
ema50 <- EMA(Cl(data), n=50)[,1]
macd <- MACD(Cl(data))[,'macd']
macd.signal <- MACD(Cl(data))[,'signal']
rsi <- RSI(Cl(data))
cmo <- CMO(Cl(data))
dc.high <- DonchianChannel(Cl(data))[,'high']
dc.low <- DonchianChannel(Cl(data))[,'low']
adx <- ADX(data)[,'ADX']
```

```
#определим модель, здесь важно сдвинуть либо предсказываемые значения на день вперед
model <- specifyModel(Next(Delt(Cl(data))) ~ atr + ema7 + ema14 + ema50 + macd + macd.signal + rsi + cmo + dc.high + dc.low + adx)

#разделим данные на две части - тренировочные и тестовые
train.window <- c('2005-01-01','2009-01-01')
validate.window <- c('2009-01-01', '2011-06-29')

train <- na.omit(as.data.frame(modelData(model, data.window=train.window)))
validate <- na.omit(as.data.frame(modelData(model, data.window=validate.window)))

#мы хотим покупать или продавать, когда индекс изменяется как минимум на 0.5% в ту или иную сторону, остальные случаи нас не интересуют
signals<-function(x) {

        if(x >= -0.005 && x<=0.005) { result<-"none" } else
        if(x > 0.005) { result<-"buy" } else
        if(x < -0.005) { result<-"sell" }

        result
}

#классифицируем имеющиеся данные, не забыв при этом выбросить из них сами цены, чтобы алгоритм не подсматривал ;
class<-sapply(train$Next.Delt.Cl.data,signals)
train<-cbind(train,class)
train$Next.Delt.Cl.data <- NULL

class<-sapply(validate$Next.Delt.Cl.data,signals)
validate<-cbind(validate,class)
validate$Next.Delt.Cl.data <- NULL
```

Теперь когда у нас готова модель данных мы можем запустить наш алгоритм для анализа. На данном шаге мы выберем некоторые значения параметров произвольно:

```
#подготовка данных закончилась, можно запускать SVM на тренировочных данных
sv<-svm(class~.,train,gamma=0.01,cost=5,kernel="radial")

#а теперь посмотрим, насколько точно наш алгоритм предсказывает тестовые данные
table(actual=validate$class,predicted=predict(sv,newdata=validate,type="class"))
```

Полученные результаты оказываются лишь немного лучше, чем бросание монетки:

```
predicted
actual  buy none sell
  buy   60  136    3
  none  39  227    3
  sell  52  102    4
```

Проведем оптимизацию параметров. Модуль e1071 содержит метод tune который применит кросс-валидацию. В процессе кросс-валидации тренировочные данные разбиваются на две части, модель тренируется на одной части, а тестируется на другой и так многократно. В итоге, выбираются лучшие параметры и модель с лучшими параметрами тренируется снова уже на полном наборе тренировочных данных.

```
#попробуем подобрать оптимальные параметры для алгоритма методом кросс-валиадации, эта строчка может выполняться несколько часов перебирая все возможные комбинации параметров

tuned <- tune(svm, class~., data = train,ranges = list(gamma = c(0.0001,0.001,0.05,0.1,0.2,0.3), cost = c(1,5,10,20,50,100,120,130)),tunecontrol = tune.control(sampling = "cross"),cross=3)

tuned
```

Дальше мы просто копируем найденные оптимальные значения и запускаем наш алгоритм снова

```
#подготовка данных закончилась, можно запускать SVM на тренировочных данных
```

```
sv<-svm(class~.,train,gamma=0.05,cost=10,kernel="radial")

#а теперь посмотрим, насколько точно наш алгоритм предсказывает тестовые данные
table(actual=validate$class,predicted=predict(sv,newdata=validate,type="class"))
```

Результаты стали намного лучше.

```
predicted
actual buy none sell
  buy   73  123    3
  none  46  215    8
  sell  48   99   11
```

Метод стохастического градиента

Пусть задана обучающая выборка $X^\ell = \{(x_i, y_i)\}_{i=1}^{\ell}, \ x_i \in \mathbb{R}^n, \ y_i \in \{-1, 1\}$.

Требуется найти вектор весов w∈ Rn, при котором достигается минимум аппроксимированного эмпирического риска:

$$Q(w, X^\ell) = \sum_{i=1}^{\ell} \mathscr{L}\big(\langle w, x_i\rangle\, y_i\big) \to \min_w$$

.

Применим для минимизации Q(w) метод градиентного спуска. В этом методе выбирается некоторое начальное приближение для вектора весов w, затем запускается итерационный процесс, на каждом шаге которого вектор wизменяется в направлении наиболее быстрого убывания функционала Q. Это направление противоположно вектору градиента : $Q'(w) = \big(\frac{\partial Q(w)}{\partial w_j}\big)_{j=1}^{n}$

w:= w− ηQ′(w),

где η>0 — величина шага в направлении антиградиента, называемая также темпом обучения (learning rate). Предполагая, что функция потерь L дифференцируема, распишем градиент:

.

$$w := w - \eta \sum_{i=1}^{\ell} \mathscr{L}'\big(\langle w, x_i\rangle\, y_i\big) x_i y_i$$

Алгоритм метода **градиентного спуска** состоит из следующих этапов:

1 этап. Задать начальную точку $\bar{x}^0$, погрешности расчета число $0<\varepsilon<1$, $\varepsilon_1>0$, $\varepsilon_2>0$, M - предельное число итераций. Найти градиент функции в произвольной точке $\nabla f(\bar{x})=\left(\frac{\partial f(\bar{x})}{\partial x_1},\ldots,\frac{\partial f(\bar{x})}{\partial x_n}\right)^T$.

2 этап. Принять $k=0$.

3 этап. Вычислить $\nabla f(\bar{x}^k)$.

4 этап. Проверить выполнение критерия окончания $\left\|\nabla f(\bar{x}^k)\right\|<\varepsilon_1$:

А) если критерий выполнен, расчет закончен $\bar{x}^*=\bar{x}^k$;

Б) если критерий не выполнен, то перейти к этапу 5;

5 этап. Проверить выполнение неравенства $k\geq M$:

А) если неравенство выполнено, то расчет окончен: $\bar{x}^*=\bar{x}^k$;

Б) если нет, то перейти к этапу 6.

6 этап. Задать величину шага t_k.

7 этап. Вычислить $\bar{x}^{k+1}=\bar{x}^k-t_k\nabla f(\bar{x}^k)$.

8 этап. Проверить выполнение условия $f(\bar{x}^{k+1})-f(\bar{x}^k)<0$.

А) если условие выполнено, то перейти к этапу 9;

Б) если условие не выполнено, принять $t_k=\frac{t_k}{2}$ и перейти к этапу 7.

9 этап. Проверить выполнение условий $\left\|\bar{x}^{k+1}-\bar{x}^k\right\|\leq\varepsilon_2$, $\left\|f(\bar{x}^{k+1})-f(\bar{x}^k)\right\|\leq\varepsilon_2$:

А) если оба условия выполнены при текущем значении k и $k=k-1$, то расчет окончен и $\bar{x}^*=\bar{x}^{k+1}$;

Б) если хотя бы одно из условий не выполнено, то принять $k=k+1$ и перейти к этапу 3.

Алгоритм метода **наискорейшего градиентного спуска** состоит из следующих этапов.

1 этап. Задать начальную точку $\bar{x}^0$, погрешности расчета $\varepsilon_1>0$, $\varepsilon_2>0$, M - предельное число итераций. Найти градиент функции в произвольной точке $\nabla f(\bar{x})=\left(\frac{\partial f(\bar{x})}{\partial x_1},\ldots,\frac{\partial f(\bar{x})}{\partial x_n}\right)^T$.

2 этап. Принять $k=0$.

3 этап. Вычислить $\nabla f(\bar{x}^k)$.

4 этап. Проверить выполнение критерия окончания $\left\|\nabla f(\bar{x}^k)\right\|<\varepsilon_1$:

А) если критерий выполнен, расчет закончен $\bar{x}^*=\bar{x}^k$;

Б) если критерий не выполнен, то перейти к этапу 5;

5 этап. Проверить выполнение неравенства $k \geq M$:

А) если неравенство выполнено, то расчет окончен: $\bar{x}^* = \bar{x}^k$;
Б) если нет, то перейти к этапу 6.

6 этап. Задать величину шага t_k^* из условия:

$$\varphi(t_k) = f(\bar{x}^k - t_k \nabla f(\bar{x}^k)) \to \min_{t_k}.$$

7 этап. Вычислить $\bar{x}^{k+1} = \bar{x}^k - t_k^* \nabla f(\bar{x}^k)$.

8 этап. Проверить выполнение условий

$$\left\| \bar{x}^{k+1} - \bar{x}^k \right\| \leq \varepsilon_2, \quad \left\| f(\bar{x}^{k+1}) - f(\bar{x}^k) \right\| \leq \varepsilon_2$$

А) если оба условия выполнены при текущем значении k и $k = k - 1$, то расчет окончен и $\bar{x}^* = \bar{x}^{k+1}$;

Б) если хотя бы одно из условий не выполнено, то принять $k = k + 1$ и перейти к этапу 3.

Пример компьютерной реализации алгоритма на языке программирования высокого уровня C++:

```
#ifndef GRADIENTSP_H
#define GRADIENTSP_H

#define NMAX 1000

#include<cmath>
#include<cstdio>
#include<iostream>

//здесь записывается целевая  функция для минимизации
double f(double x,double y)
{
   return x*x + 2*x*y + 3*y*y - 2*x -3*y;
}

//Это первая производная по dx
double f_dx(double x,double y)
{
```

```
    return 2*x + 2*y -2;
}

//Это первая производная по dy
double f_dy(double x,double y)
{
    return 2*x + 6*y -3;
}

//Это функция g в методе наискорейшего (градиентного) спуска
double g(double x, double y, double alpha)
{
    return f(x - alpha*f_dx(x,y), y - alpha*f_dy(x,y));
}

//двумерная норма
double norma(double x, double y)
{
    return sqrt(x*x+y*y);
}

//Метод половинного деления для нахождения минимума в градиентном
спуске
double Dihotomia(double a0, double b0, double epsilon, double x, double y)
{
    //Номер шага
    int k;
    //Отклонени от середины отрезка влево, вправо
    double lk, mk;
    //Величина на которую мы отклонимся от середины отрезка
    double delta=0.5*epsilon;
    //Точка минимума
    double x_;
    //Отрезок локализации минимума
    double ak=a0, bk=b0;
    k=1;
    //Пока длина отрезка больше заданной точности
    do
    {
        //Берем середину (ну почти середину - +\- некоторое дельта в частно-
сти у нас delta=0.5*epsilon)
```

```
        lk=(ak+bk-delta)/2;
        mk=(ak+bk+delta)/2;

        k++;
        //Проверяем в какую часть попадает точка минимума слева от разбиения или справа и выбираем соответствующую точку
        if(g(x,y,lk)<=g(x,y,mk))
        {
           //Теперь правая граница отрезка локализации равна mk
           bk=mk;
        }
        else
        {
           //Теперь левая граница отрезка локализации равна mk
           ak=lk;
        }
    } while ((bk-ak)>=epsilon);

    x_=(ak+bk)/2; //minimum point

    return x_;
}


// метод наискорейшего спуска
double GreatDescent(int bx, int by,double epsilon)
{
    double x[NMAX];
    double y[NMAX];
    double alpha[NMAX];
    int k;

    //Начальное приближение u[0]
    x[0]=bx;
    y[0]=by;

    std::cout <<"Результаты:"
          <<std::endl<< "x("<<0<<"): ("<<x[0]<<", "<<y[0]<<")"<<std::endl;

    for (k=0; ; k++)
    {
```

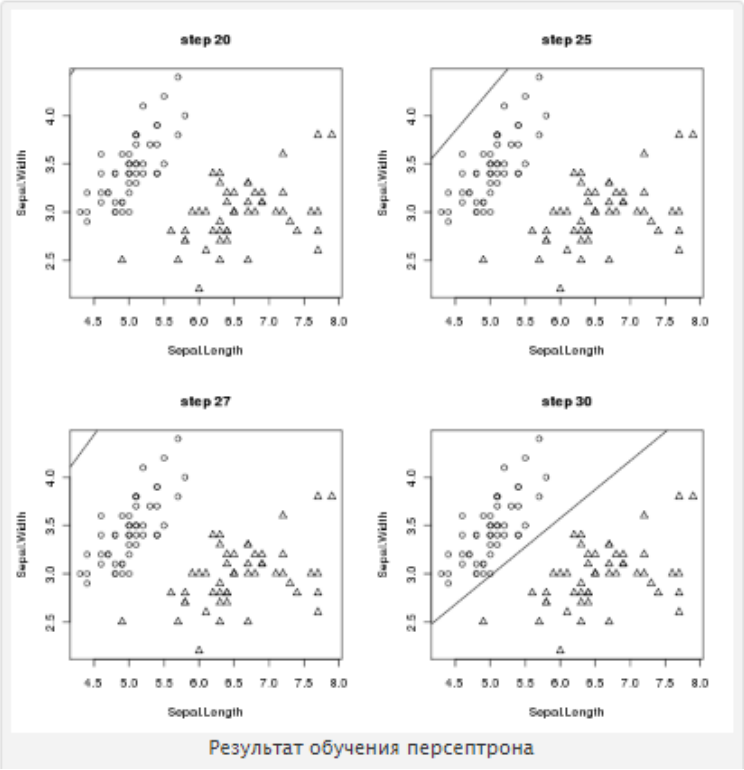
```
        //Находим alpha_k как минимум функции g на отрезке -10000,100000
        alpha[k]=Dihotomia(-10000,100000,epsilon,x[k],y[k]);
        //Вычисляем u[k]
        x[k+1]=x[k]-alpha[k]*f_dx(x[k], y[k]);
        y[k+1]=y[k]-alpha[k]*f_dy(x[k], y[k]);

        std::cout<<"x("<<k+1<<"): "<<"("<<x[k+1]<<",
"<<y[k+1]<<")"<<std::endl
                  <<"f("<<x[k+1]<<", "<<y[k+1]<<") = "<<f(x[k+1], y[k+1])
<<std::endl;
        if (k>1)
        {
           //Проверяем условие остановки
           if (norma(x[k+1]-x[k],y[k+1]-y[k])<epsilon)
           {
              break;
           }
        }
    }

    std::cout <<"Точка минимума (epsilon="<<epsilon<<")"<<std::endl
           <<"f("<<x[k+1]<<", "<<y[k+1]<<") = "<<f(x[k+1], y[k+1])
<<std::endl;
    return f(x[k+1],y[k+1]);
}
#endif // GRADIENTSP_H
```

Также для построения линейного классификатора можно использовать алгоритм персептрона

```
## Перцептрон Розенблатта
## Собственно алгоритм обучения
trainPerceptron <- function(dm, dc) {
  ## в result будем хранить список весов в зависимости от итерации,
  ## чтобы потом можно было посмотреть, как развивались события
  result <- list()
  oldW <- c()
  w <- rep(0, ncol(dm))
  while (!identical(w, oldW)) {
    oldW <- w
    for (i in 1:nrow(dm)) {
      ## %*% здесь означает скалярное произведение
      pred <- sign(dm[i,] %*% w)
      w = w + (dc[i] - pred) * dm[i, ]
    }
    ## Добавляем w в конец списка
    result <- c(result, list(w))
  }
  result
}

## Визуализация результата обучения pcResult в разные моменты времени steps
plotPerceptronSteps <- function(pcResult, dm, dc, steps) {
  for (s in steps) {
    w <- pcResult[[s]]
    plot(dm[,1:2], main=paste("step", s),
      pch=ifelse(dc > 0, 1, 2), sub=paste(w, collapse=' '))
    abline(-w[3] / w[2], -w[1] / w[2])
  }
}
```

Пример использования этой конструкции:

```
## загружаем ирисы — они, как и многие другие
## стандартные наборы данных, идут в комплекте с R
data(iris)

## Отбираем только интересующие нас классы и переводим их в матрицу dm
dd <- iris[iris$Species != "versicolor", -(3:4)]
dm <- data.matrix(dd[, 1:2])

## А это чтобы разделяющая плоскость имела вид не
## y = ax, а y = ax + b, где подбираются и a, и b
dm <- cbind(dm, 1)

## dc — правая часть классификатора (метки классов)
## Делаем вектор из единиц
dc <- rep(1, nrow(dm))
## и исправляем те, что в классе -1
dc[dd$Species == "virginica"] <- -1


## Результат обучения:
pc <- trainPerceptron(dm, dc)
plotPerceptronSteps(pc, dm, dc, length(pc))

## раскомментируйте для сохранения картинки в файл:
# png("perceptron-biased.png", width=600, height=600)

## Посмотрим разделение в разные мометны времени
par(mfrow=c(2,2))
plotPerceptronSteps(pc, dm, dc, c(20, 50, 250, length(pc)))

## раскомментируйте, если сохраняли картинку в файл:
# dev.off()
```



Результат обучения персептрона

Задания к лабораторной работе

Вариант 1

1. Создайте фрейм данных из $N = 25$ записей со следующими полями: $Nrow$ – номер записи, $Name$ – имя сотрудника, $BirthYear$ – год рождения, $EmployYear$ – год приема на работу, $Salary$ – зарплата. EyEColor – цвет глаз, SkinColor – цвет кожи, BloodType – группа крови, HairColor – цвет волос на голове. Заполните данный фрейм данными так, что $Nrow$ изменяется от 1 до N,

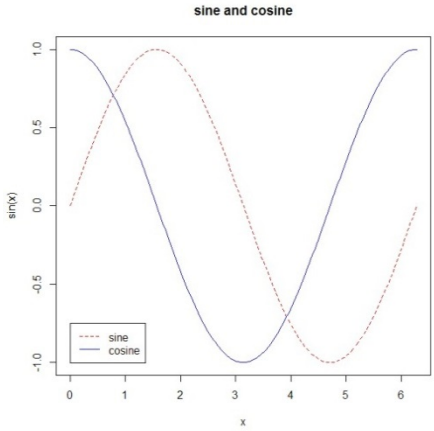


Рис. 1:

Name задается произвольно, $BithYear$ распределен равномерно (случайно) на отрезке $[1978,1995]$, $EmployYear$ распределен равномерно на отрезке $[BirthYear+18,2015]$, $Salary$ для работников младше 1990 г.р. определяется по формуле $Salary = (ln(2016 - EmployYear) + 1) * 9000$, для остальных $Salary = (log2(2016 - EmployYear) + 1) * 9000$.

Подсчитайте число сотрудников с зарплатой, большей 17000. Добавьте в таблицу поле, соответствующее суммарному подоходному налогу (ставка 13%), выплаченному сотрудником за время работы в организации, если его зарплата за каждый год начислялась согласно формулам для $Salary$, где вместо 2016 следует последовательно подставить каждый год работы сотрудника в организации.

2. Постройте линейный классификатор для классификации сотрудников данной международной организации (признаки классификации: группа крови, цвет волос, глаз и цвет кожи). Использовать машину опорных векторов и алгоритм персептрона. Полученные результаты сравнить.

Таблица соответствия групп крови национальностям

№	Национальность	Преобладающая группа крови	Примечание
1	Исландцы	1	

2	Англичане	1	
3	Итальянцы	1	
4	Датчане	2	
5	Немцы	2	
6	Русские	2	
7	Испанцы	2	
8	Китайцы	1	
9	Индусы	1	
10	Бирманцы	3	
11	Японцы	2	
12	Татары	2	
13	Китайцы Пекина	3	
14	Украинцы	2	
15	Португальцы	2	

Цвет глаз, кожи или волос можно закодировать определенным числом. В данном варианте использовать национальности: Украинец, Индус, Датчанин, Бирманец.

Для машины опорных векторов типа "C-classification" с линейным ядром, добейтесь нулевой ошибки сначала на обучающей выборке, а затем на тестовой, путем изменения параметра C.

3. Допустим, что решающая функция линейного классификатора в упрощенном виде выглядит так:

$$f(x_1, x_2) = (x_1 - 2)^2 + (x_2 - 5)^2$$

Найти координаты и значение функции в точке минимума методом градиентного спуска.

Вариант 2

1. Создайте фрейм данных из $N = 30$ записей со следующими полями: $Nrow$ – номер записи, $Name$ – имя сотрудника, $BirthYear$ – год рождения, $EmployYear$ – год приема на работу, $Salary$ – зарплата. EyEColor – цвет глаз, SkinColor – цвет кожи, BloodType – группа крови, HairColor – цвет волос на голове. Заполните данный фрейм данными так, что $Nrow$ изменяется от 1 до N,

   Name задается произвольно, $BithYear$ распределен равномерно (случайно) на отрезке $[1974,1993]$, $EmployYear$ распределен равномерно на отрезке $[BirthYear+17,2014]$, $Salary$ для работников

младше 1989 г.р. определяется по формуле $Salary = (ln(2015 - EmployYear) + 1) * 7000$, для остальных $Salary = (log2(2015 - EmployYear) + 1) * 7000$.

Подсчитайте число сотрудников с зарплатой, большей 16000. Добавьте в таблицу поле, соответствующее суммарному подоходному налогу (ставка 13%), выплаченному сотрудником за время работы в организации, если его зарплата за каждый год начислялась согласно формулам для $Salary$, где вместо 2015 следует последовательно подставить каждый год работы сотрудника в организации.

2. Постройте линейный классификатор для классификации сотрудников данной международной организации (признаки классификации: группа крови, цвет волос, глаз и цвет кожи). Использовать машину опорных векторов и алгоритм персептрона. Полученные результаты сравнить.

Цвет глаз, кожи или волос можно закодировать определенным числом. В данном варианте использовать национальности: Англичанин, Китаец, Немец.
Для машины опорных векторов типа "C-classification" с сигмоидальным ядром, добейтесь нулевой ошибки сначала на обучающей выборке, а затем на тестовой, путем изменения параметра C.

3. Допустим, что решающая функция линейного классификатора в упрощенном виде выглядит так:

$$f(x_1, x_2) = (x_2^2 + x_1^2 - 1)^2 + (x_1 + x_2 - 1)^2$$

Найти координаты и значение функции в точке минимума методом наискорейшего градиентного спуска.

Вариант 3

1. Создайте фрейм данных из $N = 27$ записей со следующими полями: $Nrow$ – номер записи, $Name$ – имя сотрудника, $BirthYear$ – год рождения, $EmployYear$ – год приема на работу, $Salary$ – зарплата. EyEColor – цвет глаз, SkinColor – цвет кожи, BloodType – группа крови, HairColor – цвет волос на голове. Заполните данный фрейм данными так, что $Nrow$ изменяется от 1 до N,

Name задается произвольно, $BithYear$ распределен равномерно (случайно) на отрезке $[1972,1994]$, $EmployYear$ распределен равномерно на отрезке $[BirthYear+19,2013]$, $Salary$ для работников младше 1980 г.р. определяется по формуле $Salary = (ln(2014 - EmployYear) + 1) * 8000$, для остальных $Salary = (log2(2014 - EmployYear) + 1) * 8000$.

Подсчитайте число сотрудников с зарплатой, большей 14000. Добавьте в таблицу поле, соответствующее суммарному подоходному налогу (ставка 13%), выплаченному сотрудником за время работы в организации, если его зарплата за каждый год начислялась согласно формулам для $Salary$, где вместо 2014 следует последовательно подставить каждый год работы сотрудника в организации.

2. Постройте линейный классификатор для классификации сотрудников данной международной организации (признаки классификации: группа крови, цвет волос, глаз и цвет кожи). Использовать машину опорных векторов и алгоритм персептрона. Полученные результаты сравнить.

Цвет глаз, кожи или волос можно закодировать определенным числом. В данном варианте использовать национальности: Русский, Татарин, Португалец, Японец, Итальянец.
Для машины опорных векторов типа "C-classification" с радиальным ядром, добейтесь нулевой ошибки сначала на обучающей выборке, а затем на тестовой, путем изменения параметра C.

3. Допустим, что решающая функция линейного классификатора в упрощенном виде выглядит так:

$$f(x_1, x_2) = 5 \cdot x_1^2 + 5 \cdot x_2^2 - 8 \cdot x_1 \cdot x_2$$

Найти координаты и значение функции в точке минимума методом градиентного спуска.

Вариант 4

1. Создайте фрейм данных из $N = 24$ записей со следующими полями: $Nrow$ – номер записи, $Name$ – имя пациента, $BirthYear$ – год рожде-

ния, $Employ$ – место работы, $Salary$ – зарплата. Cost – стоимость лечения, Albumin – содержание альбумина в крови, TransFerrin – содержание трансферрина в крови, Ferritin – содержание ферритина в крови. Заполните данный фрейм данными так, что $Nrow$ изменяется от 1 до N,

Name задается произвольно, $BithYear$ распределен равномерно (случайно) на отрезке $[1971,1997]$, , Cost для пациентов младше 1991 г.р. определяется по формуле Cost = $(ln(2013 - \text{Birth}Year) + 1) * 11000$, для остальных Cost = $(log2(2013 - \text{Birth}Year) + 1) * 10000$.

Ранжируйте пациентов по стоимости лечения, начиная с минимальной суммы. Добавьте в таблицу поле, соответствующее общему социальному вычету за лечение (ставка 13%), выплаченному пациенту, если стоимость лечения за каждый год начислялась согласно формулам для Cost, где вместо 2013 следует последовательно подставить каждый год нахождения пациента под наблюдением.

2. Постройте линейный классификатор для классификации пациентов по степени развития болезни Паркинсона (3 степени - бессимптомное течение, маловыраженные односторонние изменения, маловыраженные двусторонние изменения) на основе трех признаков – содержания в крови альбумина, трансферрина, ферритина. Использовать машину опорных векторов и алгоритм персептрона. Полученные результаты сравнить с использованием функций языка R - Predict и Table.

Для машины опорных векторов типа "C-classification" с линейным ядром, добейтесь нулевой ошибки сначала на обучающей выборке, а затем на тестовой, путем изменения параметра C.

3. Допустим, что решающая функция линейного классификатора в упрощенном виде выглядит так:

$$f(x_1, x_2) = x_1^3 + x_2^2 - x_1 \cdot x_2 - 2x_1 + 3x_2 - 4$$

Найти координаты и значение функции в точке минимума методом градиентного спуска.

Таблица показателей анализа крови

№	Показатель	Диапазон значений считающихся нормальными	Примечание
1	Альбумин	35-50	
2	Трансферрин	2-4	
3	Ферритин	20-250	
4	Билирубин	8,6-20,5	
5	Глобулин	40-60	
6	ГЕМОГЛОБИН	120-160	
7	ЭРИТРОЦИТЫ	3.7-5.1	
8	ЛЕЙКОЦИТЫ	4-9	
9	лимфоциты,	40.7	
10	нейтрофилы, ,	2-11	
11	Базофилы	0.3	
12	плазмоциты	3-11	
13	Эозинофилы	2.4	
14	ТРОМБОЦИТЫ	160-320	
15	ГЛЮКОЗА	3.5-6.5	
16	ОБЩИЙ БЕЛОК	60-80	
17	КРЕАТИНИН	0.18	
18	Гематокрит	36-48	

19	Ретикулоциты	2-12	
20	Моноциты	10	

Формы заболевания болезни Паркинсона: Дрожательно-ригидная, Акинетико-ригидная, Акинетическая форма, Дрожательная форма, ригидная форма. Степень заболевания характеризуется этапами от 0 до 5 (бессимптомное течение, маловыраженные односторонние изменения. маловыраженные двусторонние изменения, двусторонние изменения с появлением первых заметных двигательных ограничений, ощутимые ограничения, пациент сам не справляется, полная потеря трудоспособности и самостоятельной жизнедеятельности).

Вариант 5

1. Создайте фрейм данных из $N = 30$ записей со следующими полями: $Nrow$ – номер записи, $Name$ – имя пациента, $BirthYear$ – год рождения, $Employ$ – место работы, $Salary$ – зарплата. Cost – стоимость лечения, Monocit – содержание моноцитов в крови, Retikullocit – содержание ретилкуллоцитов в крови, Gematokrit – содержание гематокрита в крови, Creatin – содержание креатина в крови. Заполните данный фрейм данными так, что $Nrow$ изменяется от 1 до N,

Name задается произвольно, $BithYear$ распределен равномерно (случайно) на отрезке $[1975,1995]$, , Cost для пациентов младше 1988 г.р. определяется по формуле Cost = $(ln(2012 - \mathrm{Birth}Year) + 1) * 6000$, для остальных Cost = $(log2(2012 - \mathrm{Birth}Year) + 1) * 7000$.

Ранжируйте пациентов по стоимости лечения, начиная с максимальной суммы. Добавьте в таблицу поле, соответствующее общему социальному вычету за лечение (ставка 13%), выплаченному па-

циенту, если стоимость лечения за каждый год начислялась согласно формулам для Cost, где вместо 2012 следует последовательно подставить каждый год нахождения пациента под наблюдением.

2. Постройте линейный классификатор для классификации пациентов по степени развития болезни Паркинсона (4 степени - двусторонние изменения с появлением первых заметных двигательных ограничений, маловыраженные односторонние изменения, маловыраженные двусторонние изменения, полная потеря трудоспособности и самостоятельной жизнедеятельности) на основе четырех признаков – содержания в крови креатина, моноцитов, гематокрита, ретилкуллоцитов. Использовать машину опорных векторов и алгоритм персептрона. Полученные результаты сравнить с использованием функций языка R - Predict и Table.

Для машины опорных векторов типа "C-classification" с полиномиальным ядром, добейтесь нулевой ошибки сначала на обучающей выборке, а затем на тестовой, путем изменения параметра C.

3. Допустим, что решающая функция линейного классификатора в упрощенном виде выглядит так:

$$f(x_1, x_2) = 1 - 3 \cdot x_1 - 4 \cdot x_2 - 5 \cdot x_1 \cdot x_2 + 6 \cdot x_1^2 + 7 \cdot x_2^2$$

Найти координаты и значение функции в точке минимума методом градиентного спуска.

Вариант 6

1. Создайте фрейм данных из $N = 28$ записей со следующими полями: $Nrow$ – номер записи, $Name$ – имя пациента, $BirthYear$ – год рождения, $Employ$ – место работы, $Salary$ – зарплата. Cost – стоимость лечения, Glukosa – содержание глюкозы в крови, Trombocit – содержание тромбоцитов в крови, Belok – содержание белков в крови, Eozinofil – содержание эозинофила в крови, Plazmocit - содержание плазмоцитов в крови. Заполните данный фрейм данными так, что $Nrow$ изменяется от 1 до N,

Name задается произвольно, $BithYear$ распределен равномерно (случайно) на отрезке $[1974,1994]$, , Cost для пациентов младше 1988 г.р. определяется по формуле Cost = $(ln(2011 - \text{Birth}Year) + 1) * 9000$, для остальных Cost = $(log2(2011 - \text{Birth}Year) + 1) * 8000$.

Ранжируйте пациентов по стоимости лечения, начиная с максимальной суммы. Добавьте в таблицу поле, соответствующее общему социальному вычету за лечение (ставка 13%), выплаченному пациенту, если стоимость лечения за каждый год начислялась согласно формулам для Cost, где вместо 2011 следует последовательно подставить каждый год нахождения пациента под наблюдением.

2. Постройте линейный классификатор для классификации пациентов по формам болезни Паркинсона (5 форм - Дрожательно-ригидная, Акинетико-ригидная, Акинетическая форма, Дрожательная форма, ригидная форма) на основе пяти признаков – содержания в крови глюкозы, тромбоцитов, белков, эозинофилов, плазмоцитов. Использовать машину опорных векторов и алгоритм персептрона. Полученные результаты сравнить с использованием функций языка R - Predict и Table.

Для машины опорных векторов среди ядер "polynomial", "radial" выберите оптимальное в плане количества ошибок на тестовой выборке, для ядра типа polynomial можно изменять значение параметра degree

3. Допустим, что решающая функция линейного классификатора в упрощенном виде выглядит так:

$$f(x_1, x_2) = 70 \cdot (x_2 - x_1^2)^2 + (2 - x_1)^2$$

Найти координаты и значение функции в точке минимума методом градиентного спуска.

Вариант 7

1. Создайте фрейм данных из $N = 32$ записей со следующими полями: $Nrow$ – номер записи, $Name$ – имя пациента, $BirthYear$ – год рождения, $Employ$ – место работы, $Salary$ – зарплата. Cost –

стоимость лечения, Basifil – содержание базифилов в крови, Neitrofil – содержание нейтрофилов в крови, Limfocit – содержание лимфоцитов в крови, Leikocit – содержание лейкоцитов в крови. Заполните данный фрейм данными так, что $Nrow$ изменяется от 1 до N,

Name задается произвольно, $BithYear$ распределен равномерно (случайно) на отрезке $[1972,1992]$, , Cost для пациентов младше 1988 г.р. определяется по формуле Cost = $(ln(2013 - \mathrm{Birth}Year) + 1) * 11000$, для остальных Cost = $(log2(2013 - \mathrm{Birth}Year) + 1) * 9000$.

Ранжируйте пациентов по стоимости лечения, начиная с минимальной суммы. Добавьте в таблицу поле, соответствующее общему социальному вычету за лечение (ставка 13%), выплаченному пациенту, если стоимость лечения за каждый год начислялась согласно формулам для Cost, где вместо 2013 следует последовательно подставить каждый год нахождения пациента под наблюдением.

2. Постройте линейный классификатор для классификации пациентов по формам болезни Паркинсона (4 формы - Дрожательно-ригидная, Акинетико-ригидная, Акинетическая форма, Дрожательная форма) на основе четырех признаков – содержания в крови базифилов, нейтрофилов, лимфоцитов, лейкоцитов. Использовать машину опорных векторов и алгоритм персептрона. Полученные результаты сравнить с использованием функций языка R - Predict и Table.

Для машины опорных векторов среди ядер линейное и сигмоидальное выберите оптимальное в плане количества ошибок на тестовой и обучающей выборке

3. Допустим, что решающая функция линейного классификатора в упрощенном виде выглядит так:

$$f(x_1, x_2) = (x_1 - 2)^2 + (x_2 - 5)^2$$

Найти координаты и значение функции в точке минимума методом градиентного спуска.

Вариант 8

1. Создайте фрейм данных из $N = 27$ записей со следующими полями: $Nrow$ – номер записи, $Name$ – имя пациента, $BirthYear$ – год рождения, $Employ$ – место работы, $Salary$ – зарплата. Cost – стоимость лечения, Gemoglobin – содержание гемоглобина в крови, Limfocit – содержание лимфоцитов в крови, Bilirubin – содержание билирубина в крови. Заполните данный фрейм данными так, что $Nrow$ изменяется от 1 до N,

Name задается произвольно, $BithYear$ распределен равномерно (случайно) на отрезке $[1971,1991]$, , Cost для пациентов младше 1988 г.р. определяется по формуле Cost = $(ln(2010 - \text{Birth}Year) + 1) * 10000$, для остальных Cost = $(log2(2010 - \text{Birth}Year) + 1) * 7000$.

Ранжируйте пациентов по стоимости лечения, начиная с минимальной суммы. Добавьте в таблицу поле, соответствующее общему социальному вычету за лечение (ставка 13%), выплаченному пациенту, если стоимость лечения за каждый год начислялась согласно формулам для Cost, где вместо 2010 следует последовательно подставить каждый год нахождения пациента под наблюдением.

2. Постройте линейный классификатор для классификации пациентов по формам болезни Паркинсона (3 формы - Дрожательно-ригидная, Акинетико-ригидная, Дрожательная форма) на основе трех признаков – содержания в крови гемоглобина, лимфоцитов, билирубина. Использовать машину опорных векторов и алгоритм персептрона. Полученные результаты сравнить с использованием функций языка R - Predict и Table.

Для машины опорных векторов среди ядер полиномиальное и сигмоидальное выберите оптимальное в плане количества ошибок на тестовой и обучающей выборке (для ядра типа polynomial можно изменять значение параметра degree)

3. Допустим, что решающая функция линейного классификатора в упрощенном виде выглядит так:

$$f(x_1, x_2) = (x_1^2 + x_2 - 8)^3 + (x_1 + x_2^3 - 3)^2$$

Найти координаты и значение функции в точке минимума методом градиентного спуска.

Вариант 9

1. Создайте фрейм данных из $N = 28$ записей со следующими полями: $Nrow$ – номер записи, $Name$ – имя сотрудника, $BirthYear$ – год рождения, $EmployYear$ – год приема на работу, $Salary$ – зарплата. EyEColor – цвет глаз, SkinColor – цвет кожи, BloodType – группа крови, HairColor – цвет волос на голове. Заполните данный фрейм данными так, что $Nrow$ изменяется от 1 до N,

   Name задается произвольно, $BithYear$ распределен равномерно (случайно) на отрезке $[1974,1994]$, $EmployYear$ распределен равномерно на отрезке $[BirthYear+17,2013]$, $Salary$ для работников младше 1990 г.р. определяется по формуле $Salary = (ln(2014 - EmployYear) + 1) * 9000$, для остальных $Salary = (log2(2014 - EmployYear) + 1) * 9000$.

   Подсчитайте число сотрудников с зарплатой, большей 18000. Добавьте в таблицу поле, соответствующее суммарному подоходному налогу (ставка 13%), выплаченному сотрудником за время работы в организации, если его зарплата за каждый год начислялась согласно формулам для $Salary$, где вместо 2014 следует последовательно подставить каждый год работы сотрудника в организации.

2. Постройте линейный классификатор для классификации сотрудников данной международной организации (признаки классификации: группа крови, цвет волос, глаз и цвет кожи). Использовать машину опорных векторов и алгоритм персептрона. Полученные результаты сравнить.

   Цвет глаз, кожи или волос можно закодировать определенным числом. В данном варианте использовать национальности: Англичанин, Китаец, Датчанин, Китаец из Пекина, Исландец.
   Для машины опорных векторов типа "C-classification" с полиномиальным ядром, добейтесь нулевой ошибки сначала на обучающей выборке, а затем на тестовой, путем изменения параметра C (для ядра типа polynomial можно изменять значение параметра degree).

3. Допустим, что решающая функция линейного классификатора в упрощенном виде выглядит так:

$$f(x_1, x_2) = 3 \cdot x_1^2 + 3 \cdot x_2^3 - 9 \cdot x_1 \cdot x_2$$

Найти координаты и значение функции в точке минимума методом наискорейшего градиентного спуска.

Вариант 10

1. Создайте фрейм данных из $N = 32$ записей со следующими полями: $Nrow$ – номер записи, $Name$ – имя пациента, $BirthYear$ – год рождения, $Employ$ – место работы, $Salary$ – зарплата. Cost – стоимость лечения, Plazmocitt – содержание плазмоцитов в крови, Gemoglobin – содержание гемоглобина в крови, Neitrifil– содержание нейтрофилов в крови, Creatin – содержание креатина в крови, Gematokrit – содержание гематокрита в крови, Globulin – содержание глобулина в крови. Заполните данный фрейм данными так, что $Nrow$ изменяется от 1 до N,

Name задается произвольно, $BithYear$ распределен равномерно (случайно) на отрезке $[1972,1994]$, , Cost для пациентов младше 1986 г.р. определяется по формуле Cost = $(ln(2010 - \text{Birth}Year) + 1) * 7000$, для остальных Cost = $(log2(2010 - \text{Birth}Year) + 1) * 9000$.

Ранжируйте пациентов по стоимости лечения, начиная с максимальной суммы. Добавьте в таблицу поле, соответствующее общему социальному вычету за лечение (ставка 13%), выплаченному пациенту, если стоимость лечения за каждый год начислялась согласно формулам для Cost, где вместо 2010 следует последовательно подставить каждый год нахождения пациента под наблюдением.

2. Постройте линейный классификатор для классификации пациентов по степени развития болезни Паркинсона (3 степени - маловыраженные односторонние изменения, маловыраженные двусторонние изменения, полная потеря трудоспособности и самостоятельной жизнедеятельности) на основе шести признаков – содержания в крови плазмоцитов, гемоглобина, нейтрофилов, креатина, гематокрита, глобулина. Использовать машину опорных векторов и алгоритм персептрона. Полученные ре-

зультаты сравнить с использованием функций языка R - Predict и Table.

Для машины опорных векторов типа "C-classification" с радиальным ядром, добейтесь нулевой ошибки сначала на обучающей выборке, а затем на тестовой, путем изменения параметра C.

3. Допустим, что решающая функция линейного классификатора в упрощенном виде выглядит так:

$$f(x_1, x_2) = 5 - 4 \cdot x_1 - 3 \cdot x_2 - 2 \cdot x_1 \cdot x_2 + 4 \cdot x_1^3 + 8 \cdot x_2^3$$

Найти координаты и значение функции в точке минимума методом градиентного спуска.

Таблица характеристик полезных ископаемых – цветных металлов

№	Полезные ископаемые	Энергия ионизации атома, ЭВ	Относительная электроотрицательность	Радиус атома, НМ
1	Церий (редкоземельные)	5.6	1.12	0.181
2	Лантан (редкоземельные)	5.61	1.1	0.187
3	Самарий (редкоземельные)	5.4	1.17	0.181
4	Европий (редкоземельные)	5.47	1.2	0.185
5	Неодим (редкоземельные)	5.32	1.14	0.182
6	Тербий (редкоземельные)	5.69	1.2	0.18

7	Гадолиний (редкоземельные)	5.94	1.2	0.179
8	Диспрозий (редкоземельные)	5.67	1.2	0.18
9	Эрбий (редкоземельные)	5.84	1.24	0.178
10	Гольмий (редкоземельные)	5.74	1.23	0.179
11	Лютеций (редкоземельные)	5.13	1.27	0.175
12	Иттрий (редкоземельные)	6.15	1.22	0.178
13	Тулий (редкоземельные)	5.89	1.25	0.177
14	Тербий (редкоземельные)	5.69	1.2	0.18
15	Бериллий (легкие цветные металлы)	8.98	1.67	0.112
16	Барий (легкие цветные металлы)	5.21	0.89	0.222
17	Цинк (тяжелые)	9.06	1.65	0.138
18	Олово (тяжелые)	7.34	1.96	0.162
19	Титан (легкие)	6.58	1.54	0.147
20	Литий (легкие)	5.39	0.98	0.155
21	Рубидий (легкие)	4.17	0.82	0.248
22	Цезий (легкие	3.89	0.79	0.267

	цветные металлы)			

Вариант 11

1. Создайте фрейм данных из $N = 29$ записей со следующими полями: $Nrow$ – номер записи, $Name$ – место нахождение цветного металла, $BirthYear$ – год обнаружения, Count –прогнозируемое количество, $Salary$ – стоимость добычи. Cost – стоимость перевозки, PowerIOn – значение энергии ионизации, Electro – относительная электроотрицательности, Radius – радиус атома. Заполните данный фрейм данными так, что $Nrow$ изменяется от 1 до N,

Name задается произвольно, $BithYear$ распределен равномерно (случайно) на отрезке $[1971,1997]$, , Cost для ископаемых найденных до 1989 г.р. определяется по формуле Cost = $(ln(2006 - \mathrm{Birth}Year) +1) * 45000$, для остальных Cost = $(log2(2006 - \mathrm{Birth}Year) + 1) * 65000$. Стоимость добычи $Salary$ = $(log2(2013 + \mathrm{Birth}Year) +1)*73000$

Ранжируйте ископаемые по стоимости добычи, начиная с максимальной суммы. Добавьте в таблицу поле, соответствующее общему социальному вычету за перевозки грузов (ставка 7%), выплаченному, если стоимость перевозок за каждый год начислялась согласно формулам для Cost, где вместо 2006 следует последовательно подставить каждый год добычи полезных ископаемых.

2. Постройте линейный классификатор для классификации 5 редкоземельных цветных металлов церия, лантана, самария, европия и неодима на основе трех признаков – Энергия ионизации атома, радиус атома, Относительная электроотрицательность. Использовать машину опорных векторов и алгоритм персептрона. Полученные результаты сравнить с использованием функций языка R - Predict и Table.

Для машины опорных векторов типа "C-classification" с линейным ядром, добейтесь нулевой ошибки сначала на обучающей выборке, а затем на тестовой, путем изменения параметра C.

3. Допустим, что решающая функция линейного классификатора в упрощенном виде выглядит так:

$$f(x_1, x_2) = [(x_2 + 1)^2 + x_1^2] \times [x_1^2 + (x_2 - 1)^2]$$

Найти координаты и значение функции в точке минимума методом градиентного спуска.

Вариант 12

1. Создайте фрейм данных из $N = 33$ записей со следующими полями: $Nrow$ – номер записи, $Name$ – место нахождение цветного металла, $BirthYear$ – год обнаружения, Count –прогнозируемое количество, $Salary$ – стоимость добычи. Cost – стоимость перевозки, PowerIOn – значение энергии ионизации, Electro – относительная электроотрицательности, Radius – радиус атома. Заполните данный фрейм данными так, что $Nrow$ изменяется от 1 до N,

Name задается произвольно, $BithYear$ распределен равномерно (случайно) на отрезке $[1972,1996]$, , Cost для ископаемых найденных до 1990 г.р. определяется по формуле Cost = $(ln(2012 - \text{Birth}Year) + 1) * 55000$, для остальных Cost = $(log2(2012 - \text{Birth}Year) + 1) * 63000$. Стоимость добычи $Salary$ = $(log2(2007 + \text{Birth}Year) + 1)*74000$

Ранжируйте ископаемые по стоимости добычи, начиная с максимальной суммы. Добавьте в таблицу поле, соответствующее общему социальному вычету за перевозки грузов (ставка 8%), выплаченному, если стоимость перевозок за каждый год начислялась согласно формулам для Cost, где вместо 2012 следует последовательно подставить каждый год добычи полезных ископаемых.

2. Постройте линейный классификатор для классификации 4 редкоземельных цветных металлов тербия, гадолиния, диспрозия, эрбия на основе трех признаков – Энергия ионизации атома, радиус атома, Относительная электроотрицательность. Использовать машину опорных векторов и алгоритм персептрона. Полученные результаты сравнить с использованием функций языка R - Predict и Table.

Для машины опорных векторов типа "C-classification" с радиальным ядром, добейтесь нулевой ошибки сначала на обучающей выборке, а затем на тестовой, путем изменения параметра C.

3. Допустим, что решающая функция линейного классификатора в упрощенном виде выглядит так:

$$f(x_1, x_2) = x_1^3 + x_2^2 - x_1 \cdot x_2 - 2x_1 + 3x_2 - 4$$

Найти координаты и значение функции в точке минимума методом градиентного спуска.

Вариант 13

1. Создайте фрейм данных из $N = 30$ записей со следующими полями: $Nrow$ – номер записи, $Name$ – место нахождение цветного металла, $BirthYear$ – год обнаружения, Count –прогнозируемое количество, $Salary$ – стоимость добычи. Cost – стоимость перевозки, PowerIOn – значение энергии ионизации, Electro – относительная электроотрицательности, Radius – радиус атома. Заполните данный фрейм данными так, что $Nrow$ изменяется от 1 до N,

Name задается произвольно, $BithYear$ распределен равномерно (случайно) на отрезке $[1973,1999]$, , Cost для ископаемых найденных до 1990 г.р. определяется по формуле Cost = $(ln(2008 - \text{Birth}Year) + 1) * 57000$, для остальных Cost = $(log2(2008 - \text{Birth}Year) + 1) * 67000$. Стоимость добычи $Salary$ = $(log2(2005 + \text{Birth}Year) + 1)*77000$

Ранжируйте ископаемые по стоимости добычи, начиная с максимальной суммы. Добавьте в таблицу поле, соответствующее общему социальному вычету за перевозки грузов (ставка 11%), выплаченному, если стоимость перевозок за каждый год начислялась согласно формулам для Cost, где вместо 2008 следует последовательно подставить каждый год добычи полезных ископаемых.

2. Постройте линейный классификатор для классификации 5 редкоземельных цветных металлов гольмия, лютеция, иттрия, тулия, тербия на основе трех признаков – Энергия ионизации атома, радиус атома, Относительная электроотрицательность. Использовать машину опорных векторов и алгоритм персептрона. Полученные результаты сравнить с использованием функций языка R - Predict и Table.

Для машины опорных векторов типа "C-classification" с полиномиальным ядром, добейтесь нулевой ошибки сначала на обучающей выборке, а затем на тестовой, путем изменения параметра C. Изменяя значение параметра gamma, продемонстрируйте эффект переобучения, выполните при этом визуализацию разбиения пространства признаков на области

3. Допустим, что решающая функция линейного классификатора в упрощенном виде выглядит так:

$$f(x_1, x_2) = (x_2^2 + x_1^2 - 1)^2 + (x_1 + x_2 - 1)^2$$

Найти координаты и значение функции в точке минимума методом градиентного спуска.

Вариант 14

1. Создайте фрейм данных из $N = 27$ записей со следующими полями: $Nrow$ – номер записи, $Name$ – место нахождение цветного металла, $BirthYear$ – год обнаружения, Count –прогнозируемое количество, $Salary$ – стоимость добычи. Cost – стоимость перевозки, PowerIOn – значение энергии ионизации, Electro – относительная электроотрицательности, Radius – радиус атома.

Заполните данный фрейм данными так, что $Nrow$ изменяется от 1 до N,

Name задается произвольно, $BithYear$ распределен равномерно (случайно) на отрезке $[1970,1994]$, , Cost для ископаемых найденных до 1977 г.р. определяется по формуле Cost = $(ln(2009 - \mathrm{Birth}Year) +1) * 59000$, для остальных Cost = $(log2(2009 - \mathrm{Birth}Year) + 1) * 69000$. Стоимость добычи $Salary$ = $(log2(2007 + \mathrm{Birth}Year) +1)$*79000

Ранжируйте ископаемые по стоимости добычи, начиная с минимальной суммы. Добавьте в таблицу поле, соответствующее общему социальному вычету за перевозки грузов (ставка 10%), выплаченному, если стоимость перевозок за каждый год начислялась согласно формулам для Cost, где вместо 2009 следует последовательно подставить каждый год добычи полезных ископаемых.

2. Постройте линейный классификатор для классификации тяжелых и легких цветных металлов цинка, олова, бериллия, титана, лития, рубидия на основе трех признаков – Энергия ионизации атома, радиус атома, Относительная электроотрицательность. Использовать машину опорных векторов и алгоритм персептрона. Полученные результаты сравнить с использованием функций языка R - Predict и Table.

Для машины опорных векторов типа "C-classification" с сигмоидальным ядром, добейтесь нулевой ошибки сначала на обучающей выборке, а затем на тестовой, путем изменения параметра C. Изменяя значение параметра gamma, продемонстрируйте эффект переобучения, выполните при этом визуализацию разбиения пространства признаков на области

3. Допустим, что решающая функция линейного классификатора в упрощенном виде выглядит так:

$$f(x_1, x_2) = 5 \cdot x_1^2 + 5 \cdot x_2^2 - 8 \cdot x_1 \cdot x_2$$

Найти координаты и значение функции в точке минимума методом градиентного спуска.

Вариант 15

1. Создайте фрейм данных из $N = 31$ записей со следующими полями: $Nrow$ – номер записи, $Name$ – место нахождение цветного металла, $BirthYear$ – год обнаружения, Count –прогнозируемое количество, $Salary$ – стоимость добычи. Cost – стоимость перевозки, PowerIOn – значение энергии ионизации, Electro – относительная электроотрицательности, Radius – радиус атома. Заполните данный фрейм данными так, что $Nrow$ изменяется от 1 до N,

Name задается произвольно, $BithYear$ распределен равномерно (случайно) на отрезке $[1974,1999]$, , Cost для ископаемых найденных до 1977 г.р. определяется по формуле Cost = $(ln(2010 - \text{Birth}Year) +1) * 54000$, для остальных Cost = $(log2(2010 - \text{Birth}Year) + 1) * 64000$. Стоимость добычи $Salary$ = $(log2(2009 + \text{Birth}Year) +1)*74000$

Ранжируйте ископаемые по стоимости добычи, начиная с минимальной суммы. Добавьте в таблицу поле, соответствующее общему социальному вычету за перевозки грузов (ставка 8%), выплаченному, если стоимость перевозок за каждый год начислялась согласно формулам для Cost, где вместо 2010 следует последовательно подставить каждый год добычи полезных ископаемых.

2. Постройте линейный классификатор для классификации тяжелых, легких и редкоземельных цветных металлов цинка, олова, бериллия, титана, лития, рубидия, лантана, самария, европия, эрбия на основе трех признаков – Энергия ионизации атома, радиус атома, Относительная электроотрицательность. Использовать машину опорных векторов и алгоритм персептрона. Полученные результаты сравнить с использованием функций языка R - Predict и Table.

Для машины опорных векторов типа "C-classification" с полиномиальным ядром, добейтесь нулевой ошибки сначала на обучающей выборке, а затем на тестовой, путем изменения параметра C. Изменяя значение параметра gamma, продемонстрируйте эффект переобучения, выполните при этом визуализацию разбиения пространства признаков на области

3. Допустим, что решающая функция линейного классификатора в упрощенном виде выглядит так:

$$f(x_1, x_2) = x_1^3 + x_2^2 - 3 \cdot x_1 - 2 \cdot x_2 + 2$$

Найти координаты и значение функции в точке минимума методом градиентного спуска.