Метод опорных векторов

Метод 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.

Алгоритм метода опорных векторов реализован в пакете **e1071**. Рассмотрим основы работы с данным пакетом.

Обучение

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

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)

Параметры функции:

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 метода 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", "one-classification";

kernel тип используемого ядра. Допустимы следующие значения:

"*linear*" соответсвует ядру $K(x, y) =< x, y >$,

"*polynomial*" - $K(x, y) = (gamma < x, y > + coef 0)^{\text{degree}}$,

"*radial*" - $K(x, y) = \exp(-gamma \|x - y\|_2^2)$,

"*sigmoid*" - $K(x, y) = \text{th}(gamma < x, y > + coef\, 0)$;
degree, gamma, coef0 параметры функции ядра;

cost штрафной параметр *C*;

epsilon параметр функции потерь алгоритма ε-регрессии;

class.weights поименованный вектор весов для различных классов. 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 вероятности 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 соответствующих 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 значения решающей функции;

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 разделяющей функции для задачи восстановления регрессии и коэффициенты, умноженные на соответствующие значения **y**;

rho оцененное значение свободного параметра;

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.

Реализация алгоритма прогнозирования или тестирования

Для осуществления прогнозирования 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;

probability логическое значение, определяющее следует ли наряду с предсказаниями модели вычислять вероятности принадлежности рассматриваемого объекта к каждому из целевых классов;

na.action функция, определяющая действие, которое должно быть выполнено при обнаружении пропущенных значений.

Визуализация SVM-модели

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**, 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 именованный список, который определяет константные значения не визуализируемых 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**.

Примеры

1. Классификация сортов ирисов

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 разобьём выборку на две части: обучающую и тестовую. Последовательно обучим SVM-модели с линейным и радиальным ядрами. Затем 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)
```

predictionsTrain

	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

	setosa	*versicolor*	*virginica*
setosa	*25*	*0*	*0*
versicolor	*0*	*25*	*1*
virginica	*0*	*1*	*23*

```
predictionsTest = predict(svmModelLinear, dataIrisTestObjects)
table(dataIrisTest$"Species", predictionsTest)
```

predictionsTrain

	setosa	*versicolor*	*virginica*
setosa	*24*	*0*	*0*
versicolor	*1*	*18*	*1*
virginica	*0*	*8*	*23*

Полученные рисунки разбиений пространства признаков приведены на рис. 1 и рис. 2.

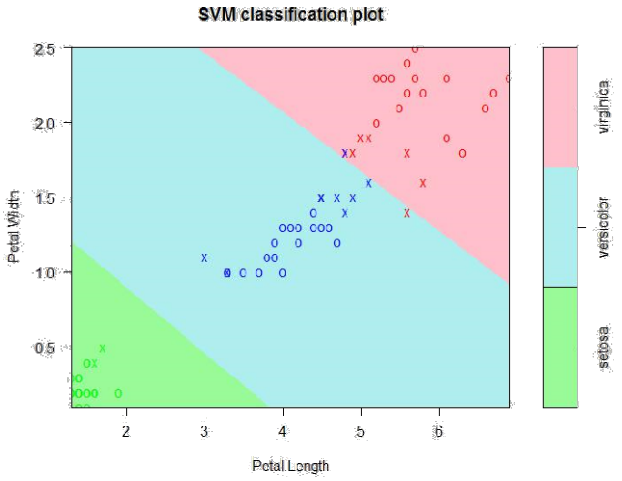

Рис. 1.

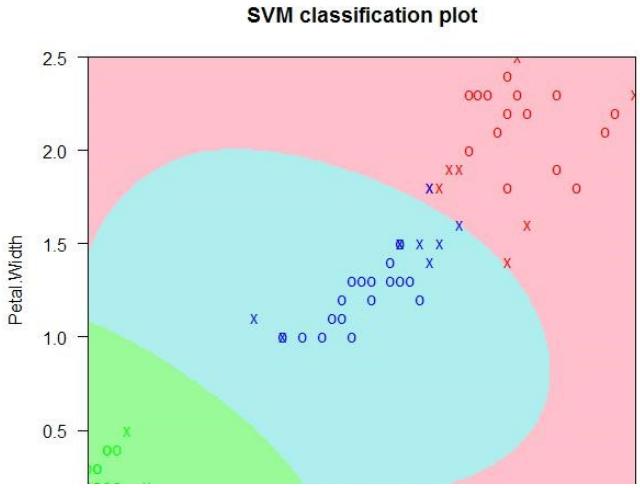

Рис. 2.

2. Восстановление регрессии

Теперь рассмотрим задачу восстановления регрессии. В качестве исследуемой зависимости вещественного значения **y** от одного вещественного признака **x** возьмем функцию $y = \log(x) + \tau$, где τ - нормальная случайная величина с математическим ожиданием 0 и дисперсией 0.3^2. Сгенерируем выборку, взяв точки $x^{(i)} = 0.1 + 0.05 \cdot i$, где $i = 1,...,98$. Построим с помощью данной выборки SVM-модель с радиальным ядром. Полученные точки обучающей выборки, опорные векторы, восстановленная зависимость и ее ε-окрестность, представлены на рис. 3.

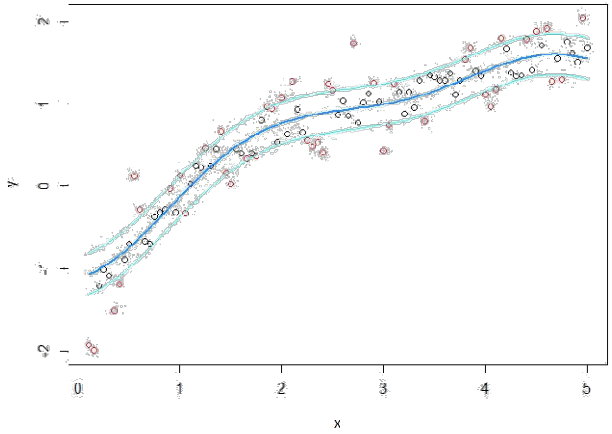

Рис. 3.

```
library ( e1071 )
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) 10 lines(x, predctions, col = "dodgerblue", lwd = 2)
lines(x, predctions + svmModel$epsilon, col = "cyan")
lines(x, predctions - svmModel$epsilon, col = "cyan")
```

Задания к лабораторной работе

Данные для обучения и тестирования SVM-моделей, которые необходимо построить в приведенных ниже заданиях, хранятся в файлах с именами svmdataI.txt и svmdataItest.txt, где I номер задания.

1. Постройте алгоритм метода опорных векторов типа "*C-classification*" с параметром C = 1, используя ядро "*linear*". Визуализируйте разбиение пространства признаков на области с помощью полученной модели. Выведите количество полученных опорных векторов, а также ошибки классификации на обучающей и тестовой выборках.
2. Используя алгоритм метода опорных векторов типа "*C-classification*" с линейным ядром, добейтесь нулевой ошибки сначала на обучающей выборке, а затем на тестовой, путем изменения параметра C. Выберите оптимальное значение данного параметра и объясните свой выбор. Всегда ли нужно добиваться минимизации ошибки на обучающей выборке?
3. Среди ядер "*polynomial*", "*radial*" и "*sigmoid*" выберите оптимальное в плане количества ошибок на тестовой выборке. Попробуйте различные значения параметра degree для полиномиального ядра.

4. Среди ядер "*polynomial*", "*radial*" и "*sigmoid*" выберите оптимальное в плане количества ошибок на тестовой выборке.
5. Среди ядер "*polynomial*", "*radial*" и "*sigmoid*" выберите оптимальное в плане количества ошибок на тестовой выборке. Изменяя значение параметра gamma, продемонстрируйте эффект переобучения, выполните при этом визуализацию разбиения пространства признаков на области.
6. Постройте алгоритм метода опорных векторов типа "*eps-regression*" с параметром C = 1, используя ядро "*radial*". Отобразите на графике зависимость среднеквадратичной ошибки на обучающей выборке от значения параметра ε. Прокомментируйте полученный результат.