МАШИННОЕ ОБУЧЕНИЕ
ЛАБОРАТОРНЫЙ ПРАКТИКУМ
(предварительный вариант)

Н.Ю. Золотых, А.Н. Половинкин

29 октября 2007 г.

1. *Практическое машинное обучение*

1.1. Метод опорных векторов

Метод опорных векторов реализован в пакете `e1071`.

1.1.1. Функция `svm` *пакета* `e1071`

Основная функция, реализующая метод — `svm`.

```
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, ..., subset, na.action =
na.omit)
```

`formula` — символическое описание исследуемой модели;

`data` — дополнительный набор данных, содержащий переменные в модели. По умолчанию переменные берутся из окружения, из которого вызывается `svm`;

`x` — матрица, вектор или разреженная матрица (объект класса `matrix.csr` реализован в пакете `SparseM`), содержащие объекты;

`y` — вектор ответов для каждой строки (компоненты) `x`. Может быть либо вектором категорий (для задач классификации), либо численным вектором (для регрессии);

`scale` — булевский вектор: может ли быть та или иная переменная масштабируема. Если вектор `scale` длины 1, то его значение распространяется на все переменные. По умолчанию, данные масштабируются внутри функции (как `x`, так и `y`) таким образом, что получить математическое ожидание, равное 0 и дисперсию, равную 1. Значения, соответствующие сдвигу и коэффициенту масштабирования возвращаются и могут быть использованы для дальнейших предсказаний;

`type` — `svm` может использоваться, как для задач классификациии, так и для восстановления регрессии или для novelty detection. В зависимости от того, является ли `y` вектором категорий или нет, значение по умолчанию для `type` – это C-классификация или ε-регрессия, соответственно. Данное значение может быть переписано указанием параметра `type` явным образом. Допустимые значения:

- C-classification

- nu-classification
- one-classification (for novelty detection)
- eps-regression
- nu-regression

`kernel` — тип ядра, используемый в обучении и предсказании. Допустимы следующие типы ядер:

- `linear`: (u, v)
- `polynomial`: $(gamma \cdot (u, v) + coef0)^{degree}$
- `radial basis`: $e^{-gamma \cdot \|u-v\|^2}$
- `sigmoid`: $\tanh(gamma \cdot (u, v) + coef0)$

`degree` — параметр, необходимый для ядра типа `polynomial` (по умолчанию равен 3);

`gamma` — параметр, необходимый для всех типов ядер, исключая `linear` (по умолчанию равен 1/(размерность данных));

`coef0` — параметр, необходимый для ядер типов `polynomial` и `sigmoid` (по умолчанию равен 0);

`cost` — цена нарушения ограничений (по умолчанию 1). Соответствует константе C в методе C-классификации.

`nu` — параметр, необходимый при использовании `nu-classification`, `nu-regression` и `one-classification`;

`class.weights` — поименованный вектор весов для различных классов (используется в случае, когда классы имеют существенно разные размеры). Не все веса должны быть обязательно указаны (по умолчанию вес класса равен 1). Все компоненты должны быть поименованы.

`cachesize` — размер кэша (по умолчанию 40 Мб);

`tolerance` — толерантность в критерии останова работы метода (по умолчанию 0.001);

`epsilon` — значение ε в функции нечувствительности (insensitive-loss function) (по умолчанию 0.1);

`shrinking` — использовать или нет эвристику сжатия (shrinking) (по умолчанию TRUE);

`cross` — если определено значение $k > 0$, то выполяется $k-$ кратная перекрестная проверка, чтобы оценить качество модели: степень точности для задачи классификации, средняя квадратичная ошибка для регрессии;

`fitted` — должны ли адаптированные значения переменных вычисляться и включаться в модель или нет (по умолчанию TRUE);

`probability` — может ли модель позволять вероятностные предсказания;

`...` — дополнительные параметры для низкоуровневой функции `svm.default`;

`subset` — индексный вектор, определяющий прецеденты, которые должны быть использованы в обучающем наборе;

`na.action` — функция, определяющая действие, которое должно быть выполнено при обнаружении отсутствующих значений (`NA`). Значение по умолчанию равно `na.omit`, что приводит к исключению прецедентов с отсутствующими значениями требуемых переменных. Альтернативой является использование `na.fail`, которая сообщает об ошибке при обнаружении отсутствующих переменных.

1.1.2. Детали

Для задач классификации с числом классов $k > 2$, `libsvm` использует последовательный подход, в котором происходит обучение $\frac{k(k-1)}{2}$ бинарных классификаторов. Соответствующий класс находится с использованием схемы голосования.

`libsvm` использует разреженное представление данных, которое также поддерживается пакетом `SparseM`.

Результатом работы функции является объект класса 'svm', содержащий обученную модель, который включает в себя:

`SV` — опорные вектора (возможно масштабированные);

`index` — индексный вектор, определяющий опорные вектора в матрице исходных данных. Данные индексы относятся к уже обработанным исходным данным (после использования `na.omit` и `subset`.

`coefs` — the corresponding coefficients times the training labels.

`rho` — the negative intercept;

`sigma` — в случае вероятностной регрессионной модели, масшабирующий параметр гипотетического $(1-0)$ распределения Лапласа, оцененный с помощью максимального правдоподобия.

`probA, probB` — числовой вектор длины $k(k-1)/2$ (k — число классов), содержащий параметры логистических распределений,...

1.1.3. Замечание

В дальшейшем будут рассматриваться задачи классификации, в которых объекты представляют собой вещественные векторы $(X1, X2) \in \mathbf{R}^2$, ответы — $Color \in \{red, green\}$.

1.1.4. 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 функию 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)
```

1.1.6. Задания к лабораторной работе

Для лабораторной работы используются заранее сгенерированные обучающие и тестовые выборки, хранящиеся в файлах `svmdataI.txt`, `svmdataItest.txt`, $I = 1, \ldots, 6$ соответственно. Каждая выборка соответствует заданию с номером I.

1) Используя ядро типа `linear`, все остальные параметры по умолчанию, исследуйте обученную модель (опорные вектора, количество ошибок классификации и т.п.).

2) Используя ядро типа `linear` и изменяя значение константы `C`, добиться 1)нулевого количества ошибок классификации на обучающей выборке; 2)нулевого количества ошибок классификации на тестовой выборке. Выберите оптимальное значение константы C, объясните свой выбор. Всегда ли нужно добиваться минимизации ошибки на обучающей выборке и почему?

3) Среди ядер типов `polynomial`, `radial`, `sigmoid` выберите оптимальное в смысле количества ошибок на тестовой выборке. Подсказка: для ядра

типа `polynomial` можно изменять значение параметра `degree`.

4) Среди ядер типов `polynomial`, `radial`, `sigmoid` выберите оптимальное в смысле количества ошибок на тестовой выборке.

5) Среди ядер типов `polynomial`, `radial`, `sigmoid` выберите оптимальное в смысле количества ошибок на тестовой выборке. Изменяя значение параметра `gamma`, продемонстрируйте эффект переобучения модели.

6) Исследуйте зависимость средней квадратической ошибки регрессии от значения параметра ε на обучающей выборке.