электронный журнал
МОЛОДЕЖНЫЙ НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК
Издатель ФГБОУ ВПО "МГТУ им. Н.Э. Баумана". Эл No. ФС77-51038.

УДК 004.58

Особенности применения алгоритма SVM для построения рекомендательной системы

Федоренко Ю. С., *студент*
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
кафедра «Системы обработки информации и управления»

Научный руководитель: Гапанюк Ю. Е., к.т.н, доцент
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
кафедра «Системы обработки информации и управления»
gapyu@bmstu.ru

1. Введение

Количество людей, работающих с Интернетом и пополняющих его различными данными, неуклонно растет с каждым годом. Уже создан колоссальный массив данных, анализируя которые можно многое узнать о предпочтениях пользователей и об их поведении. Поэтому сегодня становится все более важным создавать не просто приложения, хранящие какую-то информацию в базе данных, а делать интеллектуальные программы, способные эффективно пользоваться той информацией, которую люди накапливают ежедневно.

Это достаточно актуальная задача, решаемая различными интернет-магазинами, сервисами, рекомендующими музыку или фильмы, рекламными системами.

В основе большинства рекомендательных систем лежит один из двух методов[1, 6]:

1. Фильтрация содержимого. При фильтрации содержимого создаются профили пользователей или объектов, включающие в себя различную информацию о пользователях. На их основе производится обучение, а затем отсеиваются те данные, которые заведомо неинтересны пользователю.

2. Коллаборативная фильтрация. Системы коллаборативной фильтрации пытаются предсказать, насколько человеку понравится тот или иной продукт, на основе данных о том, как он и другие пользователи оценивали этот и другие продукты в прошлом.

Построение рекомендательной системы на практике, как и любую другую задачу машинного обучения, как правило, можно разделить на следующие этапы:

1. Изучение теоретических основ существующих алгоритмов, выбор наиболее подходящего подхода.

2. Подготовка входных данных для обучения.

3. Подбор оптимальных параметров при обучении алгоритма.

4. Оценивание качества полученных результатов.

При этом этапы 2-4 обычно повторяются несколько раз до тех пор, пока не будет получен приемлемый результат. Ниже подробно рассматривается каждый из этапов на примере задачи построения рекомендательной системы на основе фильтрации содержимого. В качестве алгоритма обучения выбран метод опорных векторов (SVM), его преимущества описаны в п. 2.8.

2. Теоретические основы метода опорных векторов.

Изучение теоретических основ используемых методов является достаточно важным условием успешного решения любой практической задачи, даже если использовать готовую реализацию того или иного метода машинного обучения. Глубокое понимание основ алгоритма, его сильных и слабых сторон позволяет наиболее правильным образом построить обучающую выборку, подобрать оптимальные параметры при обучении и проанализировать полученные результаты. Чтобы качественно выполнить каждую из этих процедур, необходимо четко понимать, как все работает изнутри, иначе процесс решения практической задачи превратится в гадание с малым шансом на успех. Поэтому необходимо рассмотреть теоретические основы метода опорных векторов, начиная со строгой постановки самой задачи классификации.

2.1. Математическая постановка задачи бинарной классификации

Имеется множество X – пространство объектов и Y – множество наименований классов. Обычно X представляет собой n-мерный вектор:

$$\forall \bar{x} \in X : \bar{x} = (x^1, ..., x^n), \text{ где } x^i \in R^n, \forall i = \overline{1,n} \qquad (1)$$

В случае фильтрации на основе содержимого X является профилем пользователя (например, может содержать информацию о музыке или фильмах, которые нравятся пользователю). Все профили необходимо разделить на две группы: такие, которым нравится данный продукт и такие, которым он не нравится. Таким образом, множество классов Y содержит в себе 2 элемента – да/нет (формально опишем их как 1 и -1). Итак, Y = {-1, 1}. Имеется целевая зависимость, y* : $X \to Y$, значения которой известны только на объектах обучающей выборки $X^l = (\overline{x_i}, y_i)_{i=1}^l, y_i = y^*(\overline{x_i})$. Требуется построить алгоритм $\alpha : X \to Y$, аппроксимирующий целевую зависимость на всем пространстве X.

При этом требуется минимизировать количество ошибок, допускаемых алгоритмом на обучающей выборке, т.е. из множества всех функций $A = \{a : X \to Y\}$ выбрать такую, для которой:

$$a^*(X^l) = \arg\min_{a \in A} \left[\sum_{i=1}^{l} 1_{|a^*(\bar{x}_i) \neq y^*(\bar{x}_i)} \right] \qquad (2)$$

2.2. Метод классификации алгоритма SVM

Суть метода SVM сводится к тому, чтобы найти разделяющую гиперплоскость, которая разделит все данные на две группы (рис. 1). Математически это можно выразить следующим образом[3]:

$$a(\bar{x}) = sign\left(\sum_{j=1}^{n} w_j x^j - b \right) = sign(\langle \bar{w}, \bar{x} \rangle - b), \qquad (3)$$

где $\bar{x} = (x^1, ..., x^n)$ - признаковое описание объекта, вектор $\bar{\omega} = (\omega_1, ..., \omega_n) \in R^n$ и порог $b \in R$ являются параметрами алгоритма, которые необходимо найти.

Фактически, уравнение $\langle \bar{\omega}, \bar{x} \rangle - b = 0$ задает гиперплоскость в n-мерном пространстве. Нетрудно видеть, что вектор ω является перпендикуляром к данной гиперплоскости, а коэффициент b характеризует её смещение.

Таким образом, цель алгоритма классификации SVM – на основе данных из обучающей выборки построить гиперплоскость, которая отделит элементы одного класса от другого, причем сделает это наиболее оптимальным образом.

2.3. Понятие оптимальной разделяющей плоскости

На рис. 1 видно, что существует множество гиперплоскостей, разделяющих данные:

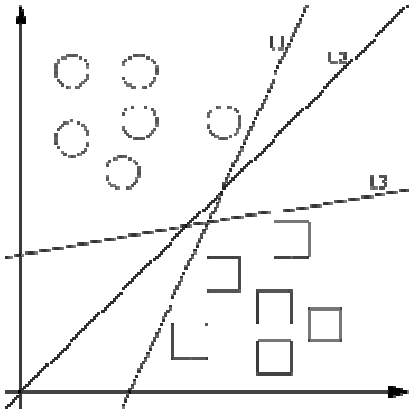

Рис. 1. Множество гиперплоскостей, разделяющих данные

Необходимо среди них выбрать оптимальную гиперплоскость. Для оценки оптимальности гиперплоскости введем понятие полосы. Способ её построения следующий: будем делать параллельный перенос исходной гиперплоскости в обе стороны, пока она не пересечет первые вектора из каждого класса. В результате получатся границы полосы (рис. 2). Ясно, что для любой гиперплоскости такая полоса может быть построена единственным образом.

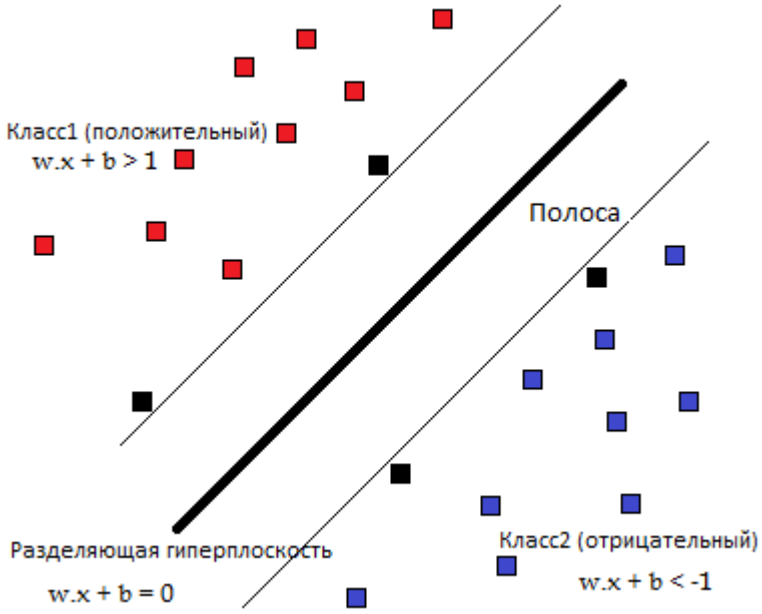

Рис. 2. Разделяющая полоса

При этом проведем линейные преобразования вектора $\bar{\omega}$ и b так, чтобы уравнения границ полосы имели вид: $\langle \bar{\omega}, \bar{x} \rangle - b = -1$ и $\langle \bar{\omega}, \bar{x} \rangle - b = 1$. Уравнение полосы тогда можно будет записать следующим образом:

$$-1 < \langle \bar{\omega}, \bar{x} \rangle - b < 1 \qquad (4)$$

Оптимальной будем считать ту гиперплоскость, для которой ширина соответствующей полосы будет максимальна (это позволит обеспечить максимальную надежность классификации в случае, если какие-то вектора окажутся внутри полосы).

А саму гиперплоскость следует расположить посередине разделяющей полосы.

Нетрудно показать[2], что ширина разделяющей полосы описывается формулой:

$$h = \frac{2}{\|\bar{\omega}\|} \qquad (5)$$

Таким образом, для достижения максимальной ширины полосы необходимо минимизировать норму вектора $\bar{\omega}$.

2.4. Постановка задачи оптимизации

Итак, построение оптимальной разделяющей гиперплоскости сводится к минимизации нормы вектора $\overline{\omega}$ при ограничениях неравенствах вида (4):

$$\begin{cases} \left\langle \overline{\omega}, \overline{\omega} \right\rangle \to \min; \\ y_i(\left\langle \overline{\omega}, \overline{x_i} \right\rangle - b) \geq 1, i = \overline{1,l} \end{cases} \qquad (6)$$

Это задача квадратичной оптимизации. Записав функцию Лагранжа и выписав необходимые условия по теореме Куна-Таккера[3], получим решение данной задачи (следует отметить, что в данном случае мы найдем локальный минимум, однако у данной функции минимум всего один в силу того, что она выпуклая):

$$\begin{cases} \overline{\omega} = \sum_{i=1}^{l} \lambda_i y_i \overline{x_i}; \\ \sum_{i=1}^{l} \lambda_i y_i = 0. \end{cases} \qquad (7)$$

Подставляя этот результат в (3), получим, что алгоритм классификации может быть записан следующим образом:

$$a(\overline{x}) = sign\left(\sum_{i=1}^{l} \lambda_i y_i \left\langle \overline{x_i}, \overline{x} \right\rangle - b \right) \qquad (8)$$

Затем останется найти коэффициенты λ, т.е. решить двойственную задачу оптимизации. Существует несколько различных алгоритмов решения данной задачи (см. п. 2.7).

2.5. Проблема шумовых выбросов. Коэффициент регуляризации

На практике может оказаться так, что обучающая выборка будет иметь шумовые выбросы (рис.3). Описанный выше классификатор провел бы узкую полосу (на рисунке она имеет синие границы). Однако такая классификация не будет оптимальной, т.к. на обучающей выборке присутствует шумовой выброс. Оптимальной может оказаться широкая полоса (с границей красного цвета).

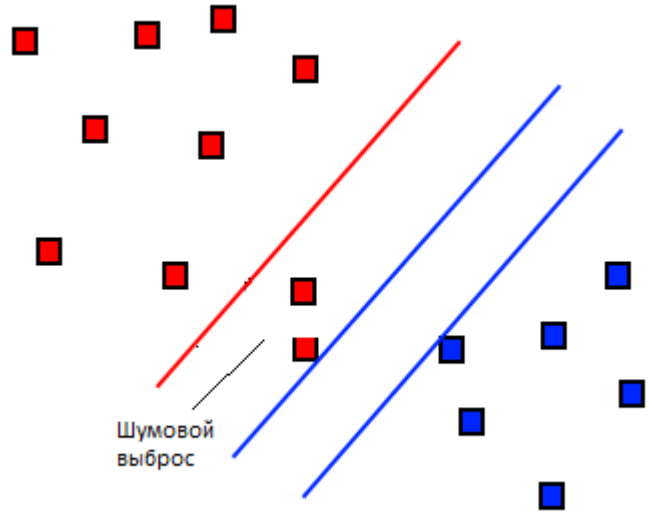

Рис. 3. Классификация выборки с шумовым выбросом

Поэтому для повышения устойчивости классификации разрешим, чтобы некоторые элементы из обучающей выборки классифицировались неправильно. При этом введем штраф за допущенную ошибку – $\varepsilon_i > 0$. Теперь будем не только минимизировать норму вектора $\overline{\omega}$, но и уменьшать суммарную величину ошибок.

Т.е. задача оптимизации (6) будет записана в следующем виде[3]:

$$\begin{cases} \frac{1}{2}\langle \overline{\omega}, \overline{\omega} \rangle + C\sum_{i=1}^{l} \varepsilon_i \to \min; \\ y_i(\langle \overline{\omega}, \overline{x_i} \rangle - b) \geq 1 - \varepsilon_i, i = \overline{1,l} \end{cases} \quad (9)$$

где ε_i – величина ошибки для k-ой точки (она равна 0 для точек, классифицировавшихся без ошибок) (рис. 4), R – количество точек, классифицировавшихся с ошибкой, C – коэффициент регуляризации. Из выражения (9) видно, что коэффициент C характеризует баланс между важностью абсолютно правильной классификации и шириной полосы. В предельном случае, когда C = 0, получится полоса бесконечной ширины, невзирая на ошибки классификации; а если C будет стремиться к бесконечности, то приоритет будет отдаваться минимальному числу ошибок. Таким образом, коэффициент регуляризации оказывает сильное влияние на процесс классификации.

Ниже изображена геометрическая интерпретация ошибки ε_i. По существу, она представляет собой расстояние до границы полосы, умноженное на норму вектора $\overline{\omega}$ (рис. 4).

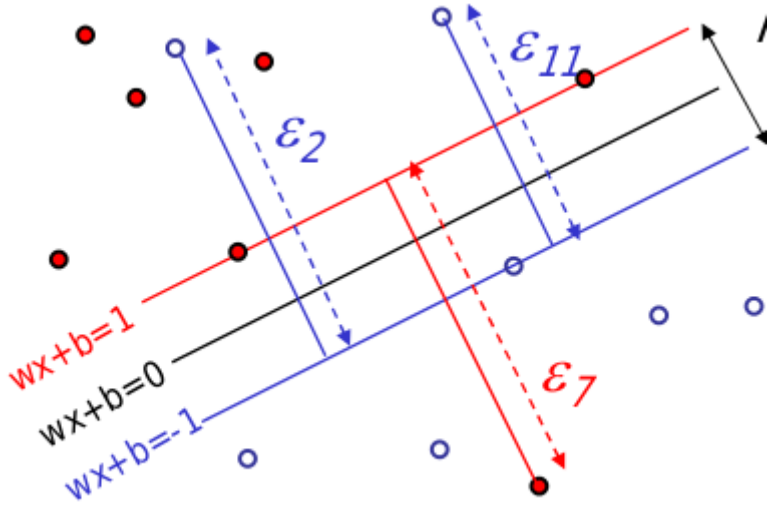

Рис. 4. Учет ошибок при классификации выборки

2.6. Проблема линейной неразделимости. Ядра

Совершенно ясно, что не всякая выборка может быть классифицирована при помощи гиперплоскости, т.е. встает проблема линейной неразделимости. Существует подход для решения данной проблемы. Это переход от исходного пространства признаков объекта X к новому пространству H с помощью некоторого преобразования $\psi : X \to H$. Если пространство H имеет достаточно высокую размерность, можно надеяться, что в нем выборка окажется разделимой.

Дадим следующее определение: функция $K : X \times X \to R$ называется ядром, если она представима в виде $K(\bar{x}, \bar{x}') = \langle \psi(\bar{x}), \psi(\bar{x}') \rangle$ при некотором отображении $\psi : X \to H$; H – пространство со скалярным произведением.

При этом для вычисления ядра необходимо скалярное произведение $\langle \bar{x}, \bar{x}' \rangle$ (8) всюду заменить скалярным произведением $\langle \psi(\bar{x}), \psi(\bar{x}') \rangle = K(\bar{x}, \bar{x}')$ в пространстве H. А $K(\bar{x}, \bar{x}')$ известно, когда ядро задано. Таким образом, вычисление ядра не представляет больших трудностей.

В выражении (8) алгоритма SVM видно, что участвуют не сами вектора $\bar{x}_i$, а их скалярные произведения классифицируемого вектора с опорными векторами. Таким образом, отслеживается «схожесть» классифицируемого вектора с опорными векторами, а затем на основе этого принимается решение. Применение ядра можно интерпретировать как изменение способа вычисления «схожести» между векторами.

2.7. Алгоритмы обучения SVM

Для решения задачи квадратичной оптимизации (6) применяются специальные методы. В принципе, способы решения квадратичных задач оптимизации известны, но они достаточно трудоемки, поэтому для обучения SVM применяют алгоритмы, учитывающие его особенности. Можно отметить следующие алгоритмы обучения SVM[4,5]:

1. Традиционные методы решения оптимизационных задач. Именно они обычно используются для обучения искусственных нейронных сетей. Сюда относятся методы поиска седловой точки функции Лагранжа, методы Ньютона, градиентные алгоритмы (например, Kernel-Adatron). Применительно к SVM данные методы используются, когда количество обучающих примеров невелико (до 4000 – 5000).

2. Декомпозиционные методы. Они основаны на разбиении одной большой задачи на несколько подзадач. Сюда относится алгоритм «образования фрагментов», который начинает работу со случайного подмножества обучающих данных, решает эту задачу и многократно добавляет примеры, которые нарушают условия оптимальности. Существуют и более совершенные алгоритмы декомпозиции, которые разбивают задачу оптимизации на неактивную и активную («рабочую») часть. Их плюс заключается в том, что они линейны относительно количества обучающих примеров и опорных векторов. Такие алгоритмы применяются в программной реализации *SVMLight*. Один из наиболее эффективных алгоритмов из этой группы – алгоритм последовательной минимальной оптимизации – применяется в *libSVM*. Существуют и некоторые другие вариации декомпозиционных алгоритмов.

3. Третья группа методов основана на вычленении из большого набора входных данных тех векторов, которые являются опорными, т.к. именно они определяют оптимальное решение задачи SVM. Сюда относятся инкрементные методы обучения SVM, которые предполагают последовательное обучение SVM на новых данных при удалении всех предыдущих данных, кроме опорных векторов. К плюсам данного алгоритма относится быстрая возможность переобучения SVM при появлении новых данных.

2.8. Преимущества и недостатки алгоритма SVM

Метод опорных векторов является одним из самых продвинутых методов классификации[6]. Однако у него также есть определенные недостатки, из-за которых он

плохо подходит для решения определенного набора практических задач. К преимуществам метода можно отнести следующее:

1. SVM – это наиболее быстрый метод нахождения решающих функций. В реальных практических задачах объем данных очень велик, а приемлемое время принятия решения в высоконагруженных системах достаточно мало. Таким образом, время обучения, а особенно время классификации на практике может стать решающим фактором.

2. Метод сводится к решению задачи квадратичного программирования в выпуклой области, которая всегда имеет одно решение. Таким образом, SVM при обучении никогда не может попасть в локальный минимум, что нередко случается при обучении нейронных сетей или генетических алгоритмов.

3. Метод находит разделяющую полосу максимальной ширины, что позволяет осуществлять более уверенную классификацию. Также следует отметить, что SVM можно представить в виде двухслойной нейронной сети, где количество нейронов скрытого слоя определяется автоматически (оно равно количеству опорных векторов).

Однако метод имеет некоторые недостатки:

1. Неустойчивость к различным шумам во входных данных. Также он требует стандартизации данных. Поэтому для успешного использования данного метода в практических задачах, необходимо проводить очень тщательно подходить к подготовке входных данных, а также проводить их нормализацию.

2. В случае линейной неразделимости классов (а в подавляющем большинстве случаев классы линейно неразделимы) не существует общего подхода к выбору ядра и построению спрямляющего пространства. Также не всегда понятно, как наиболее правильным образом выбрать коэффициент регуляризации C.

Все вышеупомянутые недостатки очень важно иметь в виду при решении практических задач.

3. Варианты использования SVM при построении рекомендательной системы

Для построения рекомендательной системы на основе SVM можно предложить два подхода:

1. Разбить всех пользователей на группы с похожим поведением. Проанализировать поведение каждой группы за определенный период. На основании этих данных составить обучающие файлы, где будет содержаться информация, какие объявления пользователям данной группы нравятся, а какие – нет. Далее нужно обучить классификатор таким образом, чтобы он отсеивал часть объявлений, которые данной группе пользователей не

нравятся, допустив при этом как можно больше ошибок второго рода. В задаче построения рекомендательной системы риски от ошибок первого и второго рода различны. Потери в результате показа пользователю объявления, которое ему не нравится (ошибка первого рода) значительно ниже, чем потери в результате отсева объявления, которое пользователю нравится (ошибка второго рода). Поэтому важно настроить классификатор таким образом, чтобы он как можно меньше терял объявлений, которые нравятся пользователям. Это достигается путем соответствующей установки штрафов и коэффициента регуляризации. Подробнее про оценку работы классификатора написано в разделе 5.

2. Возможно использование SVM в качестве одного звена рекомендательной системы, поскольку данный алгоритм не просто относит образец к тому или иному классу, но и позволяет оценить вероятность его принадлежности каждому из классов. Таким образом, если в системе оценивается вероятность того, что пользователю понравится то или иное объявление, можно использовать SVM для учета каких-либо дополнительных параметров (например, списка игр, в которые играет пользователь).

Однако при любом подходе требуется разбить всех имеющихся пользователей на группы с похожим поведением. Это можно сделать на основе каких-либо известных данных с момента регистрации (например, пол, возраст, регион и т.д.), либо на основе поведения пользователей. В последнем случае объявления, которые оценивали пользователи, откладываются в виде координатных осей. Далее в этой системе координат откладываются точки, соответствующие людям. Затем проводится кластеризация на основе какой-либо метрики (например, евклидовой).

Отдельной важной задачей является формирование обучающей и тестовой выборки. Если она не будет построена максимально корректным образом, результат получится согласно с принципом «мусор на входе – мусор на выходе». В обучающую выборку должны входить те данные, в правильности которых мы больше всего уверены. Например, это объявления, которые наиболее сильно нравятся пользователю. Кроме того, при использовании SVM достаточно важно проводить нормализацию данных. Так, установка координат всех входных векторов на отрезке [0,1] позволяет значительно улучшить конечные результаты[7].

4. Обучение алгоритма. Подбор параметров. Перекрестная проверка

На практике при обучении любого классификатора огромное значение имеет подбор его параметров. В случае алгоритма SVM при использовании ядра RBF (радиально-

базисные функции – наиболее распространенный тип ядра) необходимо осуществить подбор как минимум двух коэффициентов – коэффициента регуляризации C и параметра ядра γ. Поскольку оценить оптимальное значение этих параметров математически невозможно, на практике прибегают к эмпирической процедуре – перекрестная проверка (cross-validation или кросс-валидация). Суть данной процедуры заключается в следующем[7]:

1. Задается диапазон, в пределах которого будут меняться значения параметров C и γ . В результате получается «сетка», узлы которой задают набор пар значений C и γ, среди которых необходимо выбрать оптимальный.

2. Для каждого значения C и γ обучающая выборка разделяется на n частей (число n обычно определяется настройками), из которых n-1 часть используется для обучения при данных значениях C и γ, а одна часть – для тестирования. Данная процедура повторяется n раз (для получения более точного результата), в итоге каждая из n частей используется для тестирования; затем полученные значения усредняются. Процедура повторяется для всех значений параметра C и γ. Иногда вначале проводят кросс-валидацию с крупным шагом C и γ, а затем в том промежутке, где были достигнуты наилучшие результаты, проводят кросс-валидацию еще раз, но с меньшим шагом.

Таким образом, один цикл кросс-валидации включает разбиение набора данных на части, затем построение модели на одной части (называемой тренировочным набором), и валидация модели на другой части (называемой тестовым набором). Чтобы уменьшить разброс результатов, разные циклы перекрестной проверки проводятся на разных разбиениях, а результаты валидации усредняются по всем циклам.

Недостатком данного способа подбора параметров является большая вычислительная сложность и, как следствие, долгое время выполнения.

При проведении кросс-валидации необходимо учитывать несколько моментов[8]. Иногда кросс-валидацию используют неправильно. В этом случае ошибка предсказания на реальном наборе данных, скорее всего, будет намного хуже, чем ожидается по результатам кросс-валидации.

При проведении данной процедуры нельзя допускать следующие ошибки:

1. Использовать кросс-валидацию на нескольких моделях, и брать только результаты лучшей модели.

2. Проводить начальный анализ для определения наиболее информативного набора параметров, используя полный набор данных. Если отбор параметров требуется в модели

предсказания, он должен проводиться последовательно на каждом тренировочном наборе данных.

3. Позволять, чтобы некоторые тренировочные данные попадали также и в тестовый набор (это может случиться из-за существования дублирующих наблюдений в исходном наборе).

5. Оценка результатов бинарной классификации. Статистические метрики

При подборе параметров алгоритма путем перекрестной проверки, а также при анализе результатов на тестовой выборке, необходимо уметь оценивать качество работы алгоритма. Посчитать долю правильных ответов, данных классификатором, в ряде случаев недостаточно. Например, когда образцов положительного класса всего 1-2% (пользователю нравится 1-2% всех объявлений, имеющихся в базе интернет-магазина), классификатор, считающий, что ни одно из объявлений не нравится пользователю, будет иметь точность порядка 98-99%. Однако совершенно ясно, что такой классификатор абсолютно бесполезен.

Поэтому в машинном обучении вводятся 4 статистические метрики для оценки качества работы классификатора[9, 10]. Все образцы, для которых классификатор выдал ответ, можно разделить на четыре группы:

1. True positive – положительные результаты, верно найденные классификатором (объявления, понравившиеся пользователю, и которые были правильно определены);

2. False positive – отрицательные результаты, неверно найденные классификатором как положительные (объявления, не понравившиеся пользователю, но определенные алгоритмом как понравившиеся);

3. True negative – отрицательные результаты, верно найденные классификатором (объявления, не понравившиеся пользователю, и которые были правильно определены);

4. False negative – положительные результаты, неверно найденные классификатором как отрицательные (объявления, понравившиеся пользователю, но пропущенные алгоритмом).

Данные группы можно представить в виде таблицы.

	Объявление, понравившееся пользователю	Объявление, не понравившееся пользователю
Положительный ответ классификатора	True positive	False negative (ошибка первого рода)
Отрицательный ответ классификатора	False positive (ошибка второго рода)	True negative

На основе этих данных выделяют следующие метрики:

1. Accuracy = (true positive + true negative) / (true positive + false negative + false positive + true negative) – доля правильных ответов классификатора.

2. Precision = true positive / (true positive + false positive) – доля найденных классификатором объявлений, которые нравятся пользователю.

3. Sensitivity (recall) = true positive / (true positive + false negative) – доля правильных ответов среди положительных результатов классификатора.

4. Specificity = true negative / (true negative + false positive) – доля правильных ответов среди отрицательных ответов классификатора.

Наибольшее практическое значение имеют 2 метрики: precision и recall. При настройке классификатора необходимо добиваться нахождения максимального числа объявлений, которые нравятся пользователю (т.е. увеличивать precision). Однако обычно при увеличении precision классификатор начинает делать много «ложных срабатываний» и становится малополезным (возрастает количество ошибок первого рода). Для уменьшения нежелательного эффекта требуется следить за метрикой recall. По этой причине при решении реальных задач часто используют метрику f-score:

$$fscore = 2\frac{precision \cdot recall}{precision + recall} \quad (10)$$

Данная метрика имеет огромное практическое значение, и она часто используется при проведении кросс-валидации для оценки качества обучения при каждом значении подбираемых параметров.

6. Заключение

Построение рекомендательной системы на базе методов машинного обучения является непростой задачей. Даже если известен оптимальный подход для решения той или иной проблемы, еще необходимо правильным образом построить обучающую выборку, подобрать параметры алгоритма при обучении, а также провести оценку качества результата. Каждый из этих этапов является творческой задачей, от правильности решения которой существенно зависит получаемый результат.

Список литературы

1. Berry M.W. Large scale singular value computations // International Journal of Supercomputer Applications. 1992. № 6. P. 13–49.
2. Hendrich N., Zhang J. Support Vector Machines, 2010. Available at: http://tams-www.informatik.uni-

hamburg.de/lehre/2010ss/vorlesung/Algorithmisches_Lernen/folien/svm1.pdf, accessed 21.02.2014.

3. Воронцов К.В. Лекции по методу опорных векторов, 2007. Режим доступа: http://www.ccas.ru/voron/download/SVM.pdf (дата обращения 21.02.2014).
4. Osuna E., Freund R., Girosi F. An improved training algorithm for support vector machines // Neural Networks for Signal Processing VII. IEEE Workshop. 1997. P. 276 –285.
5. Scheinberg K. An efficient implementation of an active set method for svms // Journal of Machine Learning Research. 2006. № 7. P. 2237–2257.
6. Сегаран Т. Программируем коллективный разум: пер. с англ. А. Слинкина. СПб.: Символ-Плюс, 2008. 368 с. [Segaran T. Programming Collective Intelligence. Sebastopol: O'REILLY, 2008. 368 p.].
7. Chih-Wei Hsu, Chih-Chung Chang and Chih-Jen Lin. A Practical Guide to Support Vector Classification, 2010. Available at: http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf, accessed 21.02.2014.
8. Kohavi R. A study of cross-validation and bootstrap for accuracy estimation and model selection // The Fourteenth International Joint Conference on Artificial Intelligence: proceedings. San Mateo, CA, 1995. № 2. P. 1137–1143.
9. Wikipedia, Precision and recall. Available at: http://en.wikipedia.org/wiki/Precision_and_recall, accessed 21.02.2014.
10. Wikipedia, Sensitivity and specificity. Available at: http://en.wikipedia.org/wiki/Sensitivity_and_specificity, accessed 21.02.2014.