Автоматическая классификация текстов
Лекция № 6 курса
«Алгоритмы для Интернета»

Юрий Лифшиц*

2 ноября 2006 г.

Содержание

*Законспектировала Беляева Юлия.

1. Постановка задачи, подходы и применения

1.1. Введение

В последнее десятилетие количество электронных документов резко возросло, в связи с чем возникла необходимость решения различных задач для удобной работы с ними. *Классификация текстов* (*text categorization*, далее TC) — сортировка текстовых документов по заранее определённым категориям (в противоположность *кластеризации*, где множество категорий заранее неизвестно) — одна из таких задач. Методы TC лежат на стыке двух областей — *информационного поиска* (*information retrieval*, *извлечение информации*, далее IR) и *машинного обучения* (*machine learning*, далее ML). Общие части двух этих подходов — способы представления документов и способы оценки качества TC, а различия состоят только в способах собственно поиска.

В этой лекции TC будет рассматриваться с точки зрения ML. Согласно парадигме ML, классифицирующее правило строится постепенно на основе *тренировочной коллекции*. Такой подход обеспечивает качество классификации, сравнимое с качеством классификации, производимой человеком.

1.2. Постановка задачи

Общая постановка задачи

Пусть $D = \{d_1, \ldots, d_{|D|}\}$ — множество документов, $C = \{c_1, \ldots, c_{|C|}\}$ — множество категорий, $\Phi\colon D \times C \to \{0, 1\}$ — неизвестная целевая функция, которая по паре $\langle d_i, c_j \rangle$ говорит, принадлежит ли документ d_i категории c_j (1 или T) или нет (0 или F). Задача состоит в построении функции Φ', максимально близкой к Φ.

В такой постановке задачи надо отметить два важных момента.

1. Нет никакой дополнительной информации о категориях, это просто ярлыки, которыми помечаются документы.

2. Нет никакой внешней информации о документах, например, года публикации, источника и т.д. Известна лишь та информация, которую можно извлечь из самого документа.

Коллекция классифицированных документов

Методы ML, используемые для классификации, полагаются на наличие коллекции $\Omega = \{d_1, \ldots, d_{|\Omega|}\}$ заранее отклассифицированных документов, то есть таких, для которых уже точно известно значение целевой функции. Для того, чтобы после построения классификатора можно было оценить его эффективность, Ω разбивается на две части, не обязательно равного размера.

1. *Учебная* (*training-and-validation*) коллекция TV. Классификатор строится на основании характеристик этих документов (то есть, вообще говоря, зависит от TV).

2. *Тестовая* (*test*) коллекция Te. На ней проверяется качество классификации. Документы из Te никаким образом не должны участвовать в процессе построения классификатора, иначе полученные результаты теста уже не будут соответствовать действительности.

Кроме приведённого выше разбиения, уместным бывает дополнительно разделить коллекцию TV на две: *учебную* (*training*) Tr, на основании которой классификатор строится, и *проверочную* (*validation*) Va, на основании которой оптимизируются его параметры.

Виды классификации

Выше была поставлена задача *точной классификации*. Часто используется и так называемое *ранжирование*, при котором множество значений целевой функции — это отрезок $[0, 1]$. Документ при ранжировании может относиться не к одной, а сразу к нескольким категориям с разной степенью принадлежности, то есть категории не обязательно не пересекаются между собой.

Теоретики часто предпочитают говорить о *бинарной классификации* (*binary TC*) — случае, при котором существуют только две непересекающиеся категории. К бинарной классификации сводятся многие задачи ТС. Например, классификация по множеству категорий $C = \{c_1, \ldots, c_{|C|}\}$ (при условии независимости категорий) разбивается на $|C|$ бинарных классификаций по множеству $\{c_i, \bar{c}_i\}$, где $i \in [1, |C|]$.

Кроме описанной выше документо-центрированной классификации, иногда применяют следующую классификацию: для каждой категории выдаётся список документов, к ней принадлежащих.

1.3. Где применяется автоматическая классификация текстов

Области применения ТС:

- фильтрация спама;
- сортировка новостей;
- проверка авторства;
- подбор ключевых слов;
- составление интернет-каталогов;
- контекстная реклама;
- снятие неоднозначности (автоматические переводчики);
- автоматическое аннотирование.

2. Индексация документов

2.1. Базовый подход

Текст представляется в виде мультимножества *термов* (слов). Каждому слову сопоставлено некоторое число — *вес* — характеристика встречаемости этого слова в тексте. Порядок слов, как правило, не учитывается, учитывается только частота встречаемости слова в тексте и, возможно, другие признаки, такие как «слово встретилось в заголовке», «слово выделено другим цветом» и т.д. На основании этих признаков каждому слову в тексте сопоставляется его вес. Ещё один способ представить вес слова — $TF \cdot IDF$ (TF = *term frequency*, IDF = *inversed document frequency*). IDF показывает, насколько редко слово встречается во всех документах, TF — частота встречаемости слова в конкретном документе. Иногда проводится нормализация по документу для того, чтобы сумма квадратов всех весов в нём была равна 1. Каждый текст — это вектор многомерном пространстве, координаты — номера слов, значения координат — значения весов. Размерность вектора — это количество слов, которые встречаются в документах. Из-за того, что учитываются все слова, которые когда-либо встретились в документах, вектора получаются с огромным количеством координат, большинство из которых нули.

2.2. Уменьшение размерности векторов

Существуют различные расширения базового подхода. Например, можно по-другому выбирать термы. Бывает полезным брать в качестве термов не слова, а устойчивые группы слов, учитывать словоформы, вводить дополнительные термы (например, характеризующие длину документа). Для сокращения размерности векторов можно не учитывать редкие слова, которые увеличивают размер правила, но, как правило, не несут полезной для классификатора информации. Также можно не рассматривать слишком часто встречающиеся слова, такие как предлоги, артикли и т.п. Для каждого терма можно определить его коэффициент полезности (насколько этот терм полезен для классификации). Эту характеристику можно определить, основываясь на корреляции между встречаемостью слова в документе и принадлежностью

этого документа к одной (или нескольким) из категорий. Например, аббревиатура *ИТМО* чаще встречается в документах из категории «образование», чем в документах не из этой категории, соответственно, имеет большой коэффициент полезности.

Кроме удаления лишних термов, можно ещё группировать несколько термов в один. Например, можно группировать вместе синонимы. Ещё один подход — *совстречаемость* (*cooccurrence*): объединять слова, часто встречающиеся в одном окружении. Например, в словосочетаниях «руководитель компании», «директор компании» слова «руководитель» и «директор» встречаются перед словом «компания». Поэтому их можно объединить в один искусственный терм. В общем случае для слов определяется некая метрика близости, и группы близких слов склеиваются в один терм. Вес такого терма в каждом конкретном документе рассчитывается из весов представителей группы, которые встречаются в этом документе.

Для уменьшения размерности векторов иногда используется *сингулярное разложение* (*singular value decomposition, SVD*). Что такое сингулярное разложение? Пусть A — матрица $m \times n$. Разложение $A = USV$ называется сингулярным, если U — ортогональная матрица $m \times m$, V — ортогональная матрица $n \times n$, а S — диагональная матрица $m \times n$. Для любой матрицы A можно вычислить её сингулярное разложение. Можно показать, что если S' — матрица S, в которой оставили только k наибольших чисел, то $A' = US'V$ — самое лучшее приближение матрицы A, имеющее ранг k.

Теперь допустим, что существует такой линейно независимый набор документов $D_b = \{d_{i_1} \dots d_{i_k}\}$, что все остальные вектора документов выражаются как линейная комбинация векторов этого набора. Тогда можно взять множество D_b в качестве базиса; в таком базисе у каждого документа будет уже k координат. Таким образом можно существенно сократить размерность.

Если ранг матрицы документов больше чем k, такого базиса найти нельзя. Но сингулярное разложение позволяет найти его приближённо. Для этого надо найти искомый базис D_b для приближённой матрицы A' ранга k. Каждый вектор документа, не лежащий в линейной оболочке множества D_b, можно заменить на его проекцию. Точность такого приближения будет зависеть от выбора k.

3. Построение и обучение классификатора

3.1. Выбор порога

В первой части говорилось про точную классификацию и ранжирование. Построение классификатора второго вида обычно состоит в определении функции $CSV_i \colon D \to [0, 1]$, которая для каждого документа d_j возвращает *значение принадлежности* (*categorization status value*) d_j к c_i. Построение точного классификатора можно делать двумя способами: сразу строить функцию $CSV_i \colon D \to \{T, F\}$ или сначала вычислить аналогичную ранжированию функцию $CSV_i \colon D \to [0, 1]$, а затем определить *порог* (*threshold*) τ_i такой, что $CSV_i \geqslant \tau_i$ интерпретируется как T, а $CSV_i < \tau_i$ интерпретируется как F.

Выбирать порог можно несколькими способами.

- Пропорциональный метод. Для каждой категории c_i на коллекции Tr вычисляется, какая доля документов ей принадлежит. Пороговое значение выбирается так, чтобы на Va доля документов, отнесённых к c_i, была такой же.

- Метод k ближайших категорий. Каждый документ d_i считается принадлежащим к k ближайшим категориям и соответственно этому выбирается пороговое значение. Этот метод имеет некоторые недостатки, которые хорошо демонстрирует следующий пример. Пусть есть три категории c_1, c_2 и c_3 и два документа d_1 и d_2, а функция принадлежности CSV задаётся таблицей:

Значения CSV		Категории		
		c_1	c_2	c_3
Документы	d_1	0.6	0.1	0.3
	d_2	0.1	0.1	0.2

Если для каждого документа выбирать по одной категории, то документ d_1 попадёт в категорию c_1, а документ d_2 — в категорию c_3. При этом $CSV_3(d_1) > CSV_3(d_2)$, и d_1 не лежит в c_3, а d_2 лежит, что довольно странно. Метод k ближайших категорий используется при подборе ключевых слов.

3.2. Линейный on-line классификатор

Идея этого метода проста: правило классификатора будет скалярным произведением. Пусть каждой категории c_i соответствует вектор $\vec{c_i} = \{c_{i1}, \ldots, c_{in}\}$, где n — размерность пространства документов. В качестве правила классификатора используется следующая формула:

$$CSV_i(d) = \vec{d} \cdot \vec{c_i} = \sum_j c_{ij} d_j$$

Обычно проводится нормализация так, что итоговая формула для $CSV_i(d)$ — это просто косинус угла между вектором категории $\vec{c_i}$ и вектором документа $\vec{d}$.

$$CSV_i(d) = \frac{\vec{c_i} \cdot \vec{d}}{|\vec{c_i}||\vec{d}|}$$

На примере этого классификатора становится понятно, почему так важно было сократить количество термов: необходимо было сократить размеры векторов категорий.

Как подбираются координаты вектора $\vec{c_i}$? Они выводятся в ходе обучения (обучение проводится по каждой категории независимо от остальных). В начале выбирается тривиальный вектор $\vec{c_i} = (1, \ldots, 1)$. Для каждого учебного документа применяется текущее правило. При неудаче в координаты категории, соответствующие термам «проваленного» документа, вносятся поправки. Обычно к ним просто прибавляется заранее выбранная константа $\alpha > 0$, если полученное значение $CSV_i(d)$ меньше целевого, и вычитается константа $\beta > 0$, если значение $CSV_i(d)$ больше целевого. Необходимо заметить, что полученный в результате вектор категории будет неравномерно зависеть от векторов документов из-за того, что они обрабатываются в определённом порядке.

Имеет смысл после каждой корректировки проводить тестирование получившегося правила на Va и хранить в памяти правило, которое дало лучший результат. Если с какого-то момента обучения этот результат не улучшается, значит обучение закончено, и последнее правило с лучшим результатом и есть получившийся классификатор.

На основе получившегося вектора можно судить о незначимых для классификации словах. Если предположить, что функция CSV_i принимает значения на отрезке $[-1, 1]$, где положительные значения интерпретируются как принадлежность документа категории, то после работы алгоритма веса незначимых для классификации слов окажутся близкими к нулю. Эти слова можно не индексировать.

Существуют различные вариации этого алгоритма.

- Мультипликативные поправки. Можно использовать мультипликативные поправки для корректировки весов категории, то есть умножать соответствующие веса на коэффициенты $\alpha_1 > 1$ или $0 < \alpha_2 < 1$.

- Поправки при удачной классификации. Иногда бывает полезным немного корректировать коэффициенты и при удачной классификации.

- Поправки в неактивные слова. При неудачной классификации можно вносить поправки во все слова, а не только в слова текущего документа, в частности, пропорционально уменьшать координаты по всем неиспользуемым словам и увеличить координаты по всем используемым словам. Это может свести на нет влияние слов, которые не встречаются в учебной коллекции.

Преимущество описанного выше on-line подхода в том, что обучение можно проводить и за пределами учебной коллекции TV, непосредственно при работе правила (конечно, только при наличии обратной связи, то есть знания о том, правильно ли сработал классификатор).

3.3. Метод регрессии

Метод регрессии — это линейный, не потоковый метод. Он учится сразу на всей коллекции.

Пусть T — множество термов, C — множество категорий. Тренировочная коллекция Tr представляется виде двух матриц:

– матрица документов I размера $|Tr| \times |T|$, в которой каждая строчка — это документ, а столбец — это терм;

– матрица ответов O размера $|Tr| \times |C|$, где каждая строка соответствует документу, столбец — категории, а число на пересечении i-й строки и j-го столбца — значению $CSV_j(d_i)$.

Задача алгоритма — найти такую матрицу линейных правил M, чтобы минимизировать

$$\|MI - O\|,$$

то есть M — это $\operatorname{argmin} \|MI - O\|$.

Столбцы матрицы M — это характеристические вектора категорий. Фактически, это тот же линейный классификатор, рассмотренный ранее. Отличие только в методе обучения.

Алгоритм нахождения матрицы M:

1. Для каждой категории c_i взять вектор ответов $\vec{o_i}$ и спроецировать на линейную оболочку столбцов матрицы I.
2. Разложить полученную проекцию по базису столбцов I.
3. Это разложение и будет вектором $\vec{c_i}$, характеризующим данную категорию.

3.4. ДНФ-правило

ДНФ-классификатор состоит из множества правил вида

ЕСЛИ ⟨ДНФ-формула⟩, ТО категория

ДНФ-формула (ДНФ — дизъюнктивная нормальная форма) представляет собой дизъюнкцию нескольких конъюнкций; документ принадлежит категории, если он удовлетворяет этой формуле, то есть удовлетворяет хотя бы одному члену дизъюнкции.

На начальной стадии для каждой категории c_i, состоящей из документов $\{d^i_1, \ldots, d^i_{|c_i|}\}$, выписывается следующая формула:

$$\text{ЕСЛИ } (x = d^i_1) \text{ ИЛИ } (x = d^i_2) \text{ ИЛИ } \ldots \text{ ИЛИ } (x = d^i_{|c_i|}), \text{ ТО } c_i$$

Классификатор, основанный на таком множестве формул, абсолютно правильно работает на учебной коллекции, но, во-первых, он бесполезен на других документах, во-вторых, пользоваться таким классификатором неудобно из-за большой длины правил. Поэтому проводится серия упрощений (склеивание скобок, удаление лишних посылок и т.п.), в ходе которых может пострадать точность работы на учебной коллекции, так как главное — получить осмысленную формулу для новых документов.

3.5. Нейронные сети

В данном случае классификатор представляет собой нейронную сеть, входы которой соответствуют термам, а выходы — категориям. Для того, чтобы классифицировать документ d_j, веса его термов w_{kj} подаются на соответствующие входы сети; активация распространяется по сети, и значения, получившиеся на выходах, и есть результат классификации. Типичный метод обучения такой сети — *обратное распространение ошибки* (*back propagation*). Если на одном из тренировочных документов получен неправильный ответ на одном из выходов, то ошибка распространяется обратно по сети, и веса рёбер меняются так, чтобы ошибку уменьшить.

4. Оценка качества классификации

4.1. Метрики из информационного поиска

Какие бывают ошибки

Рассмотрим случай бинарной классификации. Есть категория c и её дополнение $\bar{c}$. Классификатор говорит, принадлежит ли документ c или нет. Таблица возможных исходов:

Категория c		Принадлежность категории	
		ДА	НЕТ
Результат классификации	ДА	true positive	false positive
	НЕТ	false negative	true negative

Здесь *true positive* (далее TP) — число документов, правильно отнесённых к категории c, *false positive* (далее FP) — число документов, неправильно отнесённых к категории c; *false negative* (далее FN) и *true negative* определяются аналогично. Обычно считается, что false negative хуже, чем false positive (например, при диагностике заболеваний объявить здорового человека больным и отправить его на дополнительное обследование не так плохо, как объявить больного человека здоровым).

Две метрики

Полнота π — это доля найденных документов из категории среди всех документов этой категории.

$$\pi = \frac{TP}{TP + FN}$$

Точность ρ — это доля найденных документов из категории среди всех документов, которые отнесены к ней классификатором (насколько много из отнесённых классификатором к категории документов на самом деле к ней принадлежат).

$$\rho = \frac{TP}{TP + FP}$$

4.2. Как сравнивать два метода классификации

Выше были определены полнота и точность. Они хорошо подходят для оценки качества классификации в ходе обучения. Два разных метода можно корректно сравнивать по полноте и точности только если:

- сравнение проводится на одной и той же коллекции (одинаковые документы и категории у обоих методов);
- коллекция одинаково разбита на Tr, Va и Te;
- документы одинаково проиндексированы.

Существует и неявный способ оценки. Классификатор сравнивается с некоторым эталонным классификатором при соблюдении вышеперечисленных условий.

Итоги

В TC применяются методы двух дисциплин: ML и IR.
Классификация производится в три этапа: индексация, построение классификатора, оценка качества.
Существуют различные классификаторы: линейный, ДНФ-правило, метод регрессии, нейронные сети.

Источники

[1] Fabrizio Sebastiani. Machine Learning in Automated Text Categorization
http://nmis.isti.cnr.it/sebastiani/Publications/ACMCS02.pdf

[2] Страница курса
http://logic.pdmi.ras.ru/~yura/internet.html