Санкт-Петербургский государственный университет

Кафедра технологий программирования

Яненко Александр Сергеевич

Выпускная квалификационная работа бакалавра

Тестирование эффективности алгоритмов нелинейной классификации в задаче распознавания типа медицинского вмешательства для рефератов статей из коллекции MEDLINE

Направление 010400
Прикладная математика и информатика

Научный руководитель:
к. ф.-м. н. доцент Добрынин В. Ю.

Рецензент:
доцент Балыкина Ю. Е.

Санкт-Петербург
2016

Содержание

Аннотация

В данной работе решается задача классификации медицинских рефератов из коллекции MEDLINE по типу медицинского вмешательства. Для этого используются ряд методов машинного обучения, как линейные, так и не линейные. Рассматриваются различные представления документов и исследуется взаимодействие методов и представлений. Входные данные сильно расбалансированы, поэтому принимается попытка решить эту проблему несколькими способами.

Введение

На протяжении почти всей истории медицина была крайне эмпирической областью. Врачи принимали решения, полагаясь либо на свой опыт, либо на опыт своих коллег, современников или нет. Врачебные практики разрабатывались в какой-то мере стихийно и передавались от целителя к целителю. Отсюда появились известные всем сомнительные практики пускания крови или даже чего-либо более мрачного — лоботомии психически больных пациентов. И даже после открытия научного метода в медицине еще долгое время преобладал опыт, а не свидетельства. В последние 20 лет, однако, все большую поддержку набирает доказательная медицина — подход, при котором врач принимает решения, основываясь исключительно на данных, полученных из подтвержденно надежных медицинских исследований, а не на опыте.

Разумеется, чтобы врачу принимать решения, основываясь на исследованиям, ему нужно иметь доступ к этим исследованиям. В этом сценарии трудно переоценить полезность какой-либо поисковой системы, которая позволила бы врачу находить нужные ему исследования без каких-либо затруднений. Конечно, исследование исследованию рознь, и то, как именно проводилось исследование, имеет большое значение. Хирургу вряд ли понадобится опыт врача-диетолога, и наоборот. Таким образом, мы приходим к основной задаче этой работы.

Основная цель — научиться автоматически различать медицинские рефераты, описывающие некий врачебный опыт, по **типу медицинского вмешательства**. Для этого имеется набор документов, уже размеченных по типу вмешательства, и стоит задача применить методы машинного обучения для того, чтобы обобщить этот опыт на произвольные документы.

Глава 1. Задача

Имеется коллекция из 8057 медицинских рефератов из коллекции MEDLINE. В каждом из них речь ведется о каком-либо исследовании, в котором применялось некое медицинское вмешательство. Все вмешательства делятся на 9 **классов**:

Классы	Кол-во док-ов
Drug	3685
Other	1440
Behavioral	883
Procedure	643
Device	544
Biological	530
Diet	274
Radiation	37
Genetic	21

Задача моей выпускной работы — используя методы машинного обучения, получить *программу-классификатор*, которая на вход будет получать некий медицинский текст, а на выходе будет возвращать тип медицинского вмешательства, описываемый в данном документе.

Глава 2. Оценка качества классификатора

Для создания классификатора, очевидно, требуется некая мера его производительности. В качестве подобных мер обычно выступают Accuracy, Precision, Recall и комбинации их, как правило F-measure.

Для их определения требуется ввести несколько понятий. Введем некий бинарный классификатор C, то есть классификатор, отвечающий на вопрос, релевантен ли входной документ или нет (понятие релевантности зависит от задчи). Введем некий набор документ, для каждого из которых известно, релевантны они или нет. Введем понятия:

- True Positive (TP) — количество объектов, которые C признал **релевантным** и которые на самом деле **релевантны**
- True Negative (TN) — количество объектов, которые C признал **не релевантными** и которые на самом деле **не релевантны**.
- False Positive (FP) — количество объектов, которые C признал **релевантным** и которые на самом деле **не релевантны**
- False Negative (FN) — количество объектов, которые C признал **не релевантными** и которые на самом деле **релевантны**.

Теперь, на основе введенных понятий введем следующие меры качества классификатора:

- `Accuracy` $= \frac{TP+TN}{TP+TN+FP+FN}$. Мера Accuracy отображает долю правильных решений среди всех документов.
- `Precision` $= \frac{TP}{TP+FP}$. Мера Precision отображает долю правильных решений среди всех релевантных документов. Интуитивно — вероятность того, что если документ определен классификатором как релевантный, он действительно релевантен.
- `Recall` $= \frac{TP}{TP+FN}$. Мера Recall отображает долю правильных предсказаний от всех релеватных документов. Интуитивно — вероятность того, что действительно релеватный документ будет определен как релевантный.

До этого момента рассматривался случай бинарной классификации. В нашем же случае классов 9. Введем некий мультиклассификатор C, который для документа возвращает один из K классов. Мера Accuracy обощается естественным способом — это все еще доля правильных предсказаний среди всех предсказаний, и это все еще 1 число. Меры же Precision и Recall обобщаются на этот случай следующим образом.

Если бинарный классификатор принимает одно решение, то C принимает K решений: принадлежит ли документ к классу 1; принадлежит ли документ к классу 2; и т. д (особенность здесь в том, что лишь на один из этих вопросов C ответит "да"). Таким образом, для классификатора C можно подсчитать K значений для каждой из статистик TP, TN, FP и FN (для каждого из классов). Из них, соответственно, получается 9 мер Precision и Recall.

Однако меры Precision и Recall сами по себе не обеспечивает объективную оценку классификации в нашем конкрентном случае.

Рассмотрим Precision. В бинарном случае классификатор может возвращать то, что документ релевантен, ровно один раз, когда он точно уверен, что этот документ релевантен. Очень велика вероятность, что в этом случае классификатор действительно попадет в релевантный документ. В этом случае $\texttt{Precision} = 1$. Однако подобный классификатор, разумеется, абсолютно бесполезен.

Рассмотрим Recall. В бинарном случае классификтор может просто всегда возвращать то, что документ релевантен, тем самым сводя значения False Negative к нулю (потому что negative предсказаний нет вовсе). В этом случае $\texttt{Recall} = 1$. Подобный классификатор, опять же, не несет никакой пользы.

В качестве компромиса крайне распространена мера F, гармоническое среднее Precision и Recall:

$$F = 2 * \frac{\texttt{Precision} \times \texttt{Recall}}{\texttt{Precision} + \texttt{Recall}}$$

Далее для оценки качества классификатора будет использоваться именно F-мера.

Глава 3. Представления документов

В этой секции кратко описываются используемые способы представления документов. Все они основываются на модели *bag-of-words* — мешок слов — что означает, что каждый документ воспринимается как набор слов без какого-либо порядка.

3.1. Векторная модель (Vector Space Model)

Используемые в этом эксперименте методы классификации используют векторную модель представления объектов [4]. Это значит, что в качестве классифицируемых объектов выступают элементы какого-либо линейного пространства — векторы. Затем на основании координат этого вектора метод делает вывод о его принадлежности к некому классу. Таким образом, используемые методы разделяют пространство на K областей, где K — количество классов.

Таким образом, для того, чтобы производить классификацию на документах, требуется каким-то образом преобразовывать их в векторы в некоем линейном пространстве. Все последующие методы делают именно это.

3.2. Term Frequency

TF является одним из самых примитивных методов преобразования документов в векторы. Идея проста — размерностью пространства N является количество уникальных слов во всей коллекции. Каждый документ представляется N-мерным вектором, где каждая координата — количество вхождений соответствующего слова в документ [4]. Подобное представление может быть полезным в отдельных случаях, однако для векторной модели, как правило, используются более продвинутые представления.

3.3. Term Frequency - Inverse Document Frequency

Одним из недостатков TF является то, что словам, часто встречающимся во всех документах, придается большое значение. Например, простейший глагол "is" будет встречаться в изобилии в абсолютно всех документах и не будет нести никакой информации, однако TF придаст большое значение этому слову. TF-IDF справляется с этой проблемой, вводя вес для каждого слова, обратно зависящий от популярности слова в коллекции. Для каждого слова вводится величина DF — document frequency — которая равна количеству документов во всей коллекции, содержащих это слово. Затем вводится величина IDF — inverse document frequency — некая функция, которая монотонно убывает при увеличении DF. Затем представления TF и IDF перемножаются по каждому слову, получая представление TF-IDF [4].

Подобное представление крайне популярно в классификации текстовых документов и доказало свою надежность годами практики.

В этой работе использовался следующий вариант меры TF-IDF (M— количество документов):

$$tf\,idf = tf * \left(\log \frac{M+1}{df+1} + 1\right)$$

3.4. Latent Semantic Indexing

Несмотря на свои преимущества, у TF-IDF есть определенные недостатки. Одним из них является тот факт, что размерность получаемого пространства равняется количеству уникальных слов в коллекции документов, что может достигать весьма больших чисел (в моем случае это было ≈ 28500). С этой проблемой призван справиться метод Latent Semantic Indexing [6]. Этот метод основывается на разложении SVD. Пусть матрица C является матрицей формы $M \times N$ Разложение SVD есть следующее разложение:

$$C = U\Sigma V^T$$

где Σ — диагональная прямоугольная матрица, где числа по диагонали убывают с увеличением индекса, матрица U является ортогональной и имеет форму $M \times M$ и матрица V является ортогональной и имеет форму $N \times N$. Числа на диагонали Σ называется *сингулярными* числами. Очевидно, что их будет r, где r — ранг матрицы C. Доказано, что наилучшее приближение матрицы C некоего ранка k будет

$$C_k = U\Sigma_k V^T$$

, где Σ_k есть матрица Σ, где были обнулены все диагональные элементы, кроме первых (и таким образом наибольших) k штук.

Теперь предположим, что C — это матрица, составленная из векторов TF-IDF для всей коллекции. Тогда M — количество документов, а N — размерность пространства (то есть, в случае TF-IDF, количество слов в колллекции). Если при этом снизить ранг этой матрицы указанным выше способом до некоего ранга k, мы получим новую матрицу C_k новой формы $M \times k$. Таким образом, для каждого документа мы получим новый вектор размерности k, который мы можем использовать в классификации. Практика показывает, что подобное преобразование не только понижает размерность, но также может даже повысить качество классификации, так как отфильтровываются слова, засоряющие общий словарь и вносящие ненужную информацию.

3.5. Probabilistic Latent Semantic Indexing

LSI часто используется в практических задачах классификации документов, однако у этого подхода есть недостаток — он никак не учитывает информацию о классах, выбирая из слов наиболее важные в общем. Следующая модель, pLSI, пытается исправить этот недостаток, используя вероятностный подход [7]. Модель pLSI является графической моделью [2], и делает следующее:

- Пользователем задается количество тем T
- Вводится T латентных переменных $z_1, \ldots, z_T$
- Вероятность появления слова w_i в документе d_j моделируется следующим образом:

$$P(w_i, d_j) = \sum_{t=1}^{T} P(z_t)P(w_i|z_t)P(d_j|z_t)$$

Обучение этой модели происходит без какой-либо информации со стороны пользователя, кроме данных в представлении TF (без учителя, *unsupervised*). Во время обучения подбираются оптимальные параметры $P(w_i|z_t)$, $P(z_t)$ и $P(z_t|d_j)$.

Данная модель может использоваться для разных целей (к примеру, разбиение документов на различные темы без обучения), однако в нашем случае мы можем использовать это для синтеза новых признаков и уменьшения размерности. Вектор из t элементов $P(z_t|d_j)$ мы можем использовать как вектор признаков для документа d_j в обучении.

Глава 4. Алгоритмы классификации

Далее кратко описаны несколько алгоритмов классификации, которые использовались в дальнейших экспериментах.

4.1. Логистическая регрессия

Логистическая регрессия — сравнительно старый, но тем не менее часто довольно эффективный алгоритм классификации [3]. Получая на вход некий вектор $\vec{x}$, этот алгоритм вычисляет некую функцию плотности $p(C_k|\vec{x})$ для каждого класса C_k, а затем выбирает класс с наибольшим значением этой функции. Функции плотности подбираются таким образом, что:

$$0 = \log\left(\frac{p(C_j|\vec{x})}{p(C_k|\vec{x}}\right) = \vec{x}^T\vec{\beta_u}$$

где $\vec{\beta_u}$ — параметры модели, которые следует подоброать во время обучения. Равенство нулю означает, что в точке $\vec{x}$ плотности равны, а значит, это граница между классами. Таким образом, логистическая регрессия стремится найти такую разделяющую поверхность между каждой парой классов, что ее логарифм будет гиперплоскостью. Вероятности тогда можно выразить так:

$$p(C_k|x) = \frac{e^{\vec{x}^T\vec{\beta_k}}}{1 + \sum_{v<K} e^{\vec{x}^T\vec{\beta_v}}}$$

$$p(C_K|x) = \frac{1}{1 + \sum_{v<K} e^{\vec{x}^T\vec{\beta_v}}}$$

β_j — параметры модели, которые следует оптимизировать в процессе обучения. Очевидно, что в формуле выше $\vec{\beta_u} = \vec{\beta_j} - \vec{\beta_k}$

Так как мы имеем дело с вероятностями, мы можем оптимизировать функцию максимального правдоподобия для входных данных $X = \{\vec{x}_1, \ldots, \vec{x}_M\}$. Оптимизация напрямую невозможна и производится с помощью методов оптимизации, таких как градиентный спуск или метод

Ньютона-Рафсона.

4.2. Машины опорных векторов

Машины опорных векторов — это линейный алгоритм **бинарной** классификации, разработанный советскими учеными В. Н. Вапником и А. Я. Червоненкисом в 1963 году[5]. Несмотря на простоту решающей функции (линейная), этот метод долгое время оставался непобежденным во многих областях, где применяется машинное обучение, и лишь в последнее десятилетие стал постепенно уступать ансамблям деревьев решений и нейронным сетям.

Первоначальная задача метода опорных векторов — построение линейного разделителя между **двумя линейно разделимыми** группами точек (в случае классификации классы и есть эти группы) **с максимальным зазором**. Это значит, что метод пытается построить разделяющую гиперплоскость, максимально удаленную от всех точек входных данных. Сам линейный классификатор имеет вид:

$$f(\vec{x}) = \texttt{sign}(\vec{\beta}^T \vec{x} + b)$$

Где $\vec{\beta}$ и b — параметры модели, которые следует подбирать.

Разумеется, в реальном мире почти никогда не встречаются линейно разделимые данные, поэтому почти всегда используется модификация метод опорных векторов с нежесткой границей (soft-margin). Это значит, что некоторые точки все-таки могут заходить за разделитель, однако взвешенная сумма того, насколько они заходят за разделитель, должна быть минимальна. Формально задача определяется следующим образом[4]:

Дано обучающее множество в виде набора пар типа (объект, ответ) $\{(\vec{x}_i, y_i\}$, где $\vec{x}_i \in R^N$, $y_i \in \{-1, +1\}$ и гиперпараметр алгоритма C. Найти $\vec{\beta}$, b and $\zeta_i \geq 0$ такой, что:

- $\frac{1}{2}||\vec{\beta}|| + C \sum_i \zeta_i$ минимален
- и для всех $\{(\vec{x}_i, y_i\}$ $y_i(\vec{\beta}^T \vec{x}_i + b) \geq 1 - \zeta_i$

Используя метод Лагранжа условной оптимизации и теорему Куна-Такера, получается двойственная задача оптимизации:

Найти $\alpha_1, \ldots, \alpha_N$ такие, что:

- $\sum \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \vec{x}_i^T \vec{x}_j$ максимален
- $\sum_i \alpha_i y_i = 0$
- $0 \leq \alpha_i \leq C$ для всех $1 \leq i \leq N$

При найденных $\alpha_1, \ldots, \alpha_N$ искомые параметры выражаются как:
$\vec{\beta} = \sum \alpha_i y_i x_i$
$b = y_k(1 - \zeta_k) - \vec{\beta}^T \vec{x}_k$ для $k = arg\, max_x \alpha_k$
Получается, что решающая функция имеет вид:

$$f(\vec{x}) = \texttt{sign}(\sum \alpha_i y_i x_i^T \vec{x} + b) = \texttt{sign}(\sum \alpha_i y_i (\vec{x}_i, \vec{x}) + b)$$

α_i будет равно нулю для большинства значений i, поэтому подобная решающая функция будет вычисляться довольно быстро. Векторы $\vec{x}_i$ такие, что $\alpha_i \neq 0$, называются *опорными*.

Как видно из решающей функции, для входного значения $\vec{x}$ требуется вычислить скалярные произведения со всемы опорными векторами, которые впоследствии и используются в решающей функции. Отсюда, можно подменить скалярное произведение $(\vec{x_i}, \vec{x})$ на некоторую другую функцию $K(\vec{x_i}, \vec{x})$, которое является скалярным произведением **в другом пространстве с большей размерностью**. Таким образом метод опорных векторов позволяет спроецировать задачу классификации в более высокоразмерное пространство почти бесплатно, что часто положительно влияет на качество классификацию. Этот прием часто называют Kernel Trick[4], а саму функцию $K(\vec{x}, \vec{y})$ — kernel-функцией. При этом при проецировании разделяющих гиперплоскостей обратно в изначальное пространство, они не обязательно линейные. Таким образом, линейный метод опорных векторов обобщается на нелинейный случай с помощью Kernel Trick.

В наших экспериментах мы будем использовать ядро RBF (Radial Ba: $K(\vec{x}, \vec{y}) = \exp\left(- \frac{||\vec{x} - \vec{y}||^2}{2\sigma^2} \right)$ и стандартное линейное.

4.3. Ансамбли деревьев решений

Деревья решений — одни из самых простых классификаторов. Суть их заключается в том, что строится дерево, в каждом узле которого располагается некое условие на один из признаков входного объекта.

Классификация какого-либо объекта в дереве решений сводится к серии проверок условий в узлах. Объект стартует в корне дерева, проходит проверку и по ее итогам идет в правого или левого потомка текущего узла. Подобные операции продолжаются, пока объект не достигнет листа. В каждом листе уже записаны значения классов, которые и присваиваются всем объектам, попавшим в этот лист.

Процесс обучения дерева решений состоит по входным **размеченным** данным создать нужное количество узлов с нужными условиями в каждом из них. На каждом этапе делается решение, нужно ли разбивать узел и дальше, затем, если да, выбирается такое условие, которое максимально разделит классы.

Гиперпараметры деревьев решений могут быть разные, но часто включают в себя максимальный размер дерева и минимальное количество элементов в листе для того, чтобы его можно было разделить.

Сами по себе деревья решений почти никогда не применяются в реальных задачах машинного обучения, так как крайне склонны к переобучению и способны очень сильно меняться при незначительных изменениях в обучающих данных. Однако именно эти свойства позволяют эффективно объединять их в *ансамбли классификаторов*.

4.3.1. Random Forest

Одним из подобных ансамблей является Random Forest[1]. Суть состоит в следующем: обучается много слабых деревьев решений (с небольшой глубиной и *только на части признаков*), чьи решения потом усредняются.

Предположим, что обучающая выборка X состоит из M объектов с N признаками. Для каждого из деревьев ансамбля сначала выбирается **с возвратом** обучающая выборка $\widetilde{X}$ **размера** M из X. Это значит,

что в $\widetilde{X}$ столько же элементов, сколько и в X, и элементы в ней могут повторяться. Затем выбирается подмножество признаков размера L, которые будут использоваться при обучении дерева. Затем на полученной выборке обучается дерево с заданными гиперпараметрами.

Гиперпараметрами этого классификатора является количество деревьев и гиперпараметры для деревьев. Частое значение для количества признаков $L = \sqrt{N}$.

Несмотря на кажущуюся простоту, подобные классификаторы зачастую работают очень хорошо, не уступая, а порой и обгоняя машины опорных векторов.

4.3.2. Gradient Boosting

Другим ансамблем является метод Gradient Boosting [9]. Его основное отличие от Random Forest заключается в том, что в RF деревья строятся независимо друг от друга, в то время как GB на каждом шаге улучшает предыдущую модель.

Пусть у нас есть некие данные x, для которых известен идеальный ответ y. На каждом шаге m алгоритма GB у нас есть некая модель F_m. Существует некая идеальная модель h, которая полностью приближает текущую модель к идеалу:

$$y = F_m(x) + h(x)$$

Основная идея алгоритма Gradient Boosting состоит в том, чтобы построить некую $\hat{h}(x)$, насколько можно приближающую функцию $h(x) = y - F_m(x)$, и затем перейти к следующей итерации $F_{m+1} = F_m + \hat{h}(x)$. Функция $\hat{h}(x)$ ищется среди слабых деревьев решений.

Этот алгоритм набирал и набирает популярность с начала 2000-х годов и на сегодняшний момент во многих задачах является де-факто базовым решением.

Глава 5. Обобщение бинарного классификатора на мультиклассификатор

Существуют классификаторы, которые не обобщаются на случай, когда существует больше двух классов, как, например, SVM. В данном случае существует два основных подхода[10]:

1. One vs rest — строятся K классификаторов (по количеству классов), каждый из которых отделяет один из классов от всех остальных. Во время классификации какого-либо объекта каждый классификатор возвращает некую величину, отображающую уверенность в принадлежности к соответствующему классу. Затем выбирается класс, чей классификатор вернул наибольшую уверенность. В случае SVM мера уверенности есть дальность точки от разделяющей плоскости. В случае логистической регрессии это просто вероятность, возвращаемая решающей функцией.

2. One vs one — строятся $\frac{K(K-1)}{2}$ классификаторов, по одному на каждую пару классов. Во время классификации возвращается тот класс, которого выбрали наибольшее количество из этих классификаторов. Преимущество этого метода в том, что классификатор не должен возвращать некую меру уверенности в своем решении.

Во время экспериментов над конкретными данными в описанной задаче выяснено, что лучше работает подход one vs rest.

Глава 6. Несбалансированность данных

Посмотрим еще раз на распределение объектов по классам.

Классы	Кол-во док-ов
Drug	3685
Other	1440
Behavioral	883
Procedure	643
Device	544
Biological	530
Diet	274
Radiation	37
Genetic	21

Как видно, данные распределены по классам крайне неравномерно. Существуют различные способы борьбы с этой проблемой. Все, кроме Gradient Boosting, алгоритмы имеют возможность в какой-то мере это компенсировать, придавая больший вес объектам из классов с малым количеством объектов. Этот вес, как правило, обратно пропорционален размеру класса. Таким образом, во всех дальнейших экспериментах этот прием используется, где это возможно. Однако есть и другой прием, который будет рассмотрен позже на отобранных классификаторах.

Глава 7. Эксперименты и результаты

В следующей секции описанные методы будут применяться в поставленной ранее задаче классификации.

7.1. Эксперименты с представлением pLSA

Для иллюстрации работы pLSA сделаем следующее:

- На всей коллекции обучить модель pLSA с 50 компонентами
- Извлечь векторы $P(d_j|z_t)$
- Для каждого из классов найдем центроиду в новом представлении

$$\mu_k = \frac{1}{|C_k|} \sum_{d_j \in C_k} P(d_j|z_t)$$

- В каждой центроиде найдем максимальную компоненту

$$z_k^* = \texttt{arg}\,\texttt{max}_t P(\mu_k|z_t)$$

- Для этой компоненты выведем слова с наибольшими значениями $P(w_i|z_t)$

Таким образом, по идее, мы должны увидеть слова, относящиеся к теме. Вот результат подобного эксперимента:

Radiation	patients survival cancer stage treatment arm radiotherapy chemotherapy free breast
Drug	mg dose patients daily day adverse safety study events doses
Genetic	cell cells expression patients gene blood peripheral clinical transplantation genes
Other	trial study clinical randomized trials controlled data design treatment evidence
Diet	intake diet protein dietary acid fatty energy consumption fat food
Behavioral	participants intervention smoking adherence use self based cessation alcohol interventions
Device	stent eluting patients stents lesions target lesion coronary mm drug
Biological	vaccine dose influenza hpv vaccination years 18 antibody doses study
Procedure	surgery patients surgical postoperative complications randomized undergoing laparoscopic study procedure

Как видно, все выбранные слова очень хорошо совпадают с тематикой класса. Таким образом, можно уже предположить, что сжатие пространства признаков успешно. Однако настоящие результаты оцениваются дальше в эксперименте.

7.2. Выбор наилучшего представления данных для каждого алгоритма

Далее будут рассматриваться различные комбинации описанных представлений данных и описанных алгоритмов. Для оценивания качества классификации, как и было сказано раньше, будет использоваться F-мера:

$$F = 2 * \frac{\texttt{Precision} \times \texttt{Recall}}{\texttt{Precision} + \texttt{Recall}}$$

Будет использоваться усредненная по 5-fold кросс-валидации F-мера для каждого класса.

В качестве представлений данных используются TF-IDF (где доступно), LSI с 1000 и 100 компонентами и pLSI с 100 и 50 компонентами. Размерность векторов в случае TF-IDF — 21260.

Для каждой из комбинаций "классификатор-представление" были выбраны наиболее оптимальные значения гиперпараметров классификатора. Подбор осуществлялся с помощью поиска по решетке (Grid Search)[8].

7.2.1. SVM с линейным ядром

Классы	Размер	TF-IDF	LSI-1000	LSI-100	pLSI-100	pLSI-50
Behavioral	883	0.629018	0.636827	0.590877	0.584237	0.576048
Biological	530	0.699443	0.708856	0.693377	0.678650	0.684562
Device	544	0.633876	0.595281	0.513737	0.467320	0.462922
Diet	274	0.535768	0.526009	0.486059	0.491578	0.453839
Drug	3685	0.856870	0.842271	0.814083	0.789520	0.792017
Genetic	21	0.233333	0.279365	0.090197	0.046688	0.041077
Other	1440	0.586627	0.561314	0.484410	0.489659	0.443479
Procedure	643	0.514435	0.486494	0.460158	0.423555	0.386138
Radiation	37	0.522120	0.511967	0.430345	0.240617	0.317722

Как видно, уменьшение размерности не помогает модели SVM с линейным ядром — чем меньше размерность, тем меньше значение F-меры.

7.2.2. SVM с ядром RBF

Классы	Размер	TF-IDF	LSI-1000	LSI-100	pLSI-100	pLSI-50
Behavioral	883	0.587566	0.630977	0.616821	0.589838	0.591519
Biological	530	0.691098	0.711623	0.636990	0.624138	0.604651
Device	544	0.595434	0.615004	0.508689	0.446495	0.470814
Diet	274	0.402198	0.552495	0.467704	0.468842	0.449692
Drug	3685	0.833642	0.853046	0.772239	0.733793	0.738837
Genetic	21	0.000000	0.180952	0.117105	0.098095	0.012500
Other	1440	0.593261	0.575825	0.468588	0.458345	0.420685
Procedure	643	0.485578	0.504082	0.444991	0.404629	0.398169
Radiation	37	0.402424	0.515584	0.514564	0.342663	0.301075

Как видно, представление LSI-1000 имеет значительное преимущество над всеми остальными и производительность алгоритма в этом случае на уровне с SVM с линейным ядром с представлением TF-IDF.

7.2.3. Логистическая регрессия

Классы	Размер	TF-IDF	LSI-1000	LSI-100	pLSI-100	pLSI-50
Behavioral	883	0.629823	0.633698	0.579547	0.574158	0.563971
Biological	530	0.720685	0.720292	0.697459	0.681986	0.686369
Device	544	0.629695	0.602882	0.513400	0.441758	0.473877
Diet	274	0.540810	0.543951	0.476940	0.477417	0.453000
Drug	3685	0.852546	0.850908	0.819772	0.783337	0.796436
Genetic	21	0.257143	0.195455	0.079459	0.053733	0.052978
Other	1440	0.561903	0.550899	0.491505	0.486786	0.463375
Procedure	643	0.539933	0.513250	0.458835	0.420920	0.386519
Radiation	37	0.519824	0.543062	0.447026	0.258564	0.301051

Как видно, представления TF-IDF и LSI-1000 дают наибольшее качество предсказания. Общая производительность — на уровне с линейным SVM в комбинации с TF-IDF.

7.2.4. Gradient Boosting

В данном случае применение модели TF-IDF невозомжно, так как метод Gradient Boosting не в состоянии справиться с такой большой размерностью — обучение будет длиться слишком долго.

Классы	Размер	LSI-1000	LSI-100	pLSI-100	pLSI-50
Behavioral	883	0.588746	0.581403	0.559861	0.551091
Biological	530	0.697918	0.688260	0.671289	0.689965
Device	544	0.534609	0.548993	0.488692	0.450415
Diet	274	0.375757	0.400635	0.423086	0.402219
Drug	3685	0.832818	0.834244	0.818545	0.822947
Genetic	21	0.000000	0.080000	0.000000	0.080000
Other	1440	0.570096	0.546445	0.533512	0.514809
Procedure	643	0.433770	0.434491	0.400049	0.395539
Radiation	37	0.355902	0.411868	0.494286	0.364791

Представления явно не вносят большого различия, ни одно из них явно не доминирует над всеми остальными. В общем и целом, в нашем случае качество работы этого метода совсем невысоко — явно ниже предыдущих трех методов.

7.2.5. Random Forest

Классы	Размер	TF-IDF	LSI-1000	LSI-100	pLSI-100	pLSI-50
Behavioral	883	0.619831	0.225643	0.527751	0.539916	0.549984
Biological	530	0.666207	0.548114	0.654345	0.672947	0.693184
Device	544	0.358864	0.239332	0.413714	0.411229	0.427579
Diet	274	0.212152	0.007143	0.257371	0.336103	0.349183
Drug	3685	0.768453	0.689127	0.786826	0.796097	0.809383
Genetic	21	0.000000	0.000000	0.000000	0.000000	0.000000
Other	1440	0.637043	0.394389	0.541252	0.534889	0.520694
Procedure	643	0.208375	0.012308	0.319505	0.403690	0.359344
Radiation	37	0.222222	0.272222	0.490404	0.421966	0.450085

Самая высокое качество достигается при использовании представления pLSI-100 (хотя и не по всем классам), однако даже в этом случае качество предсказания ниже, чем у Gradient Boosting. В нашем случае этот метод работает не очень хорошо.

Глава 8. Балансировка данных

Посмотрим еще раз на распределение объектов по классам.

Классы	Кол-во док-ов
Drug	3685
Other	1440
Behavioral	883
Procedure	643
Device	544
Biological	530
Diet	274
Radiation	37
Genetic	21

Как видно, класс "Drug" сильно доминирует над всеми остальными — ближайший по численности класс больше чем вдвое меньше. Попробуем сбалансировать набор данных, убрав часть данных с меткой "Drug" и добавиви искуственных значений в остальные классы. Я решил из каждого класса **с возвратом** выбрать 2000 значений и использовать их в качестве обучающего множества. Такая техника называется **oversampling**. Посмотрим на получившиеся результаты. Используем представление LSI-1000. Вот F-мера линейного SVM, усредненная по 5-fold кросс-валидации:

Классы	F-мера	Размер
Behavioral	0.818742	2000
Biological	0.929062	2000
Device	0.870592	2000
Diet	0.933133	2000
Drug	0.779918	2000
Genetic	0.996765	2000
Other	0.668776	2000
Procedure	0.805854	2000
Radiation	0.992804	2000

Такие статистики выглядят довольно впечатляюще, однако, учитывая то, как были созданы эти данные, нельзя до конца доверять этой статистике. Возьмем другой набор рефератов, который раньше нигде не применялся. Обучим линейный SVM на искуственно сбалансированных данных и на обычных данных и сравним F-меру:

Классы	Обычные данные	Искуственно сбалансированные данные
Behavioral	0.973883740522	0.688907422852
Biological	0.961770623742	0.700729927007
Device	0.972860125261	0.577777777778
Diet	0.956962025316	0.629343629344
Drug	0.98224852071	0.803589920608
Genetic	1.0	0.666666666667
Other	0.940092165899	0.283513097072
Procedure	0.974958263773	0.547400611621
Radiation	0.971428571429	0.681818181818

Подобный подход явно не удался.

Глава 9. Выводы

Таким образом, для классификации подобных данных можно с равным успехом использовать линейный SVM с представлением TF-IDF, SVM с RBF-ядром и даже логистическую регрессию с представлением TF-IDF.

Глава 10. Заключение

Таким образом, была рассмотрена задача классификации медицинских рефератов по типу вмешательства. Были рассмотреные различные классификаторы и представления данных и определены оптимальные их сочетания для этой конкретной задачи. Была проведена попытка сбалансировать данные с помощью техники oversampling, обернувшаяся неудачей на реальных данных.

Список литературы

[1] Breiman, Leo. "Bagging predictors." Machine learning 24.2 (1996): 123-140.

[2] Bishop, Christopher M. "Pattern Recognition." Machine Learning (2006).

[3] Cosma Shalizi. Undergraduate Advanced Data Analysis. 2012. `http://www.stat.cmu.edu/~cshalizi/uADA/12/lectures/ch12.pdf`

[4] Christopher D. Manning, Prabhakar Raghavan and Hinrich Schutze. *An Introduction to Information Retrieval.* Cambridge University Press, Cambridge, England, 2009.

[5] Cortes, Corinna, and Vladimir Vapnik. "Support-vector networks." Machine learning 20.3 (1995): 273-297.

[6] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. Indexing by latent semantic analysis. Journal of the American Society for Information Science (1990).

[7] Hofmann, Thomas. "Probabilistic latent semantic indexing." Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval. ACM, 1999.

[8] scikit-learn documentation. Grid Search: Searching for estimator parameters `http://scikit-learn.org/stable/modules/grid_search.html`

[9] Ridgeway, Greg. "Generalized Boosted Models: A guide to the gbm package." Update 1.1 (2007): 2007.

[10] Rifkin, Ryan. "Multiclass Classification." Lecture Slides. February (2008).