МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Кафедра дискретной математики и алгоритмики

САТАНЕВСКИЙ
Владислав Валерьевич

АЛГОРИТМЫ ОБУЧЕНИЯ И ВЫВОДА В ГРАФИЧЕСКИХ ВЕРОЯТНОСТНЫХ МОДЕЛЯХ ДЛЯ АНАЛИЗА ДАННЫХ ОБ УСТОЙЧИВОСТИ К ЛЕКАРСТВЕННЫМ ПРЕПАРАТАМ

Дипломная работа

Научный руководитель:
профессор, доктор физико-математических наук
А.В. Тузиков

Допущен к защите
«____» _____ 2016 г.

Зав. кафедрой дискретной математики и алгоритмики
профессор …….

Белорусский государственный университет
Кафедра дискретной математики и алгоритмики

Утверждаю
Заведующий кафедрой _____ Котов Д.М.

Дата

ЗАДАНИЕ НА ДИПЛОМНУЮ РАБОТУ

Обучающемуся студенту Сатаневскому В.В.

1. Тема дипломной работы

Алгоритмы обучения и вывода в графических моделях для анализа лекарственной устойчивости к противомикробным препаратам.

Утверждена приказом ректора БГУ от _____ № _____

2. Исходные данные к дипломной работе

Liu, J. High-Dimensional Structured Feature Screening Using Binary Markov Random Fields / J. Liu [et al.]. // JMLR workshop and conference proceedings. – 2012. - №22. – P. 712–721.

Bishop, C.M. Graphical Models / C.M. Bishop // Pattern Recognition and Machine Learning. – Springer, 206. - Ch. 8. – P. 359-418.

Xiong, H.Y. The human splicing code reveals new insights into the genetic determinants of disease / H.Y. Xiong [et al.]. // Science. – 2015. - №347(6218). – P. 347-364.

Bengio, Y. Deep learning of representations: Looking forward / Y. Bengio // Statistical Language and Speech Processing. – Springer Berlin Heidelberg, 2013. – P. 1-37.

3. Перечень подлежащих разработке вопросов или краткое содержание расчетно-пояснительной записки:

а) изучение аппарата графических моделей, включая марковские случайные поля и байесовские сети, а также методов глубинного обучения (deep learning);

б) анализ применимости изученных методов к решению задачи поиска мутаций лекарственной устойчивости, построение математической модели;

в) применение выбранных методов к реальным данным (наборам полных геномов микобактерий туберкулеза), оценка результатов.

4. Перечень графического материала (с точным указанием обязательных чертежей и графиков)

5. Консультанты по дипломной работе (дипломному проекту) с указанием относящихся к ним разделов: н.с. ОИПИ НАН Беларуси Сергеев Р.С.

6. Примерный календарный график выполнения дипломной работы

- октябрь – ноябрь 2015: изучение литературы и алгоритмов;
- декабрь – февраль 2015: реализация и тестирование отобранных алгоритмов;
- февраль– апрель 2016: проведение экспериментов по обработке геномных данных микобактерий туберкулеза, оценка результатов;
- май 2016: оформление дипломной работы.

7. Дата выдачи задания - октября 2015г.

8. Срок сдачи законченной дипломной работы – июнь 2016г.

Руководитель _____ Тузиков А.В.

Подпись обучающегося _____ Дата

Реферат

Дипломная работа, ##FIXME## с., ##FIXME## рис., ##FIXME## таблиц, ##FIXME## источник, ##FIXME## приложения.

ПОЛНОГЕНОМНЫЙ АНАЛИЗ, МИКРОБАКТЕРИИ ТУБЕРКУЛЕЗА, ОТБОР ПРИЗНАКОВ, ГРАФИЧЕСКИЕ МОДЕЛИ, МАРКОВСКИЕ СЛУЧАЙНЫЕ ПОЛЯ, СЕТЬ РЕЛЕВАНТНЫХ ПРИЗНАКОВ, МЕТОД СКОЛЬЗЯЩЕГО КОНТРОЛЯ.

Объектом исследования является аппарат графических моделей на примере задачи поиска мутаций, влияющих на лекарственную устойчивость, и методы отбора признаков.

Целью работы является изучение аппарата графических моделей на примере задачи поиска мутаций, влияющих на лекарственную устойчивость. Изучение методов отбора признаков. Построение окружения, позволяющего тестировать методы отбора признаков. Применение выбранных моделей к реальным данным. Оценка результатов.

В результате исследованы графические модели на примере задачи поиска мутаций, влияющих на лекарственную устойчивость. Исследованы методы отбора признаков. Реализованы изученные методы. Построено окружение, позволяющее тестировать методы отбора признаков. Проведен сравнительный анализ методов отбора признаков.

Методы исследования: машинное обучение, графические модели.

Область применения: полногеномный анализ, определение лекарственной устойчивости.

Abstract

Graduation work, ##FIXME## p., ##FIXME## pictures, ##FIXME## tables, ##FIXME## sources, ##FIXME## appendixes.

COMPLETE GENOME ANALYSIS, MYCOBACTERIUM TUBERCULOSIS, FEATURE SELECTION, GRAPHICAL MODELS, MARKOV RANDOM FIELDS, FEATURE RELEVANCE NETWORK, CROSS-VALIDATION.

Object of research: graphical models in case of mutation search task, where mutation affects the drug resistance, feature selection methods.

Goal of research: study graphical models in case of mutation search task, where mutation affects the drug resistance, study feature selection methods, build the environment, that allows to test feature selection methods, apply the selected models to the real data, evaluate the results.

As the result of the current research, graphical models in case of mutation search task, where mutation affects the drug resistance, were studied. Feature selection methods were investigated. Investigated methods were implemented. The environment, that allows to test the feature selection methods, was built. A comparative analysis of feature selection methods was made.

Research methods: machine learning, graphical models.

Applications: complete genome analysis, drug resistance recognition.

Рэферат

Дыпломная работа ##FIXME## с., ##FIXME## рыс., ##FIXME## табліц, ##FIXME## крыніц, ##FIXME## прыкладанняў.

ПОЛНАГЯНОМНЫ АНАЛІЗ, МІКРАБАКТЭРЫЯ ТУБЕРКУЛЁЗА, АДБОР ПРЫКМЕТ, ГРАФІЧНЫЯ МАДЭЛІ, МАРКАЎСКІЯ ВЫПАДКОВЫЯ ПАЛЯ, СЕТКА РЭЛЕВАНТНЫХ ПРЫКМЕТ, МЕТАД ЗМЕННАГА КАНТРОЛЮ.

Аб’ектам даследавання з’яўляецца апарат графічных мадэлей на ўзоры задачы пошука мутацый, якія ўплываюць на лекавую ўстойлівасць, і метады адбора прыкмет.

Мэтай работы з’яўляецца вывучэнне апарата графічных мадэлей на ўзоры задачы пошука мутацый, якія ўплываюць на лекавую ўстойлівасць. Вывучэнне метадаў адбора прыкмет. Пабудова наваколля, якое дазваляе тэстіраваць метады адбора прыкмет. Скарыстанне выбраных мадэлей да рэальных даных. Аценка вынікаў.

У выніку даследаваны графічныя мадэлі на ўзоры задачы пошука мутацый, якія ўплываюць на лекавую ўстойлівасць. Даследаваны метады адбора прыкмет. Рэалізаваны даследаваныя метады. Пабудавана наваколле, якое дазваляе тэстіраваць метады адбора прыкмет. Праведзен параўнальны аналіз метадаў адбора прыкмет.

Метады даследавання: машыннае навучанне, графічныя мадэлі.

Галіна скарыстання: полнагяномны аналіз, выяўленне лекавай устойлівасці.

Оглавление

Введение

Развитие высокопроизводительных методов секвенирования дает значительный толчок биологическим исследованиям и становлению персонализированной медицины. Получение полного генетического кода живых организмов стало гораздо более доступным.

Однако, несмотря на это, использование генетического кода довольно затруднительно. Эти трудности связаны с тем, что влияние отдельных аллелей и нуклеотидов на различные процессы, происходящие в живых организмах, изучены лишь частично. В связи с этим, большое распространение получили алгоритмы машинного обучения, позволяющие автоматически находить закономерности в геноме, влияющие на исследуемые процессы. Однако и здесь есть некоторые трудности, связанные с тем, что, как правило, количество имеющихся размеченных примеров, необходимых алгоритмам машинного обучения, невелико, а количество признаков внушительно. Это затрудняет построение устойчивых моделей, не страдающих от эффекта переобучения. Также, несмотря на то, что получение полного генома стало гораздо доступней, это все еще достаточно дорогостоящая процедура. Гораздо более дешевой является процедура получения небольших участков генома. Описанные выше проблемы определяют необходимость поиска небольшого числа значимых участков генома. Если число используемых для анализа участков невелико, с одной стороны, модель будет использовать гораздо меньшее число признаков, а значит, наверняка получится более устойчивой, а с другой, вычисление этих признаков будет дешевле.

В данной работе рассматриваются некоторые способы отбора значимых признаков и оценка влияния отобранных признаков на развитие резистентности к противомикробным препаратам.

ГЛАВА 1. ОСНОВНЫЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

1.1 Постановка задачи

В данной работе рассматривается задача полногеномного поиска ассоциаций, где анализируются мутации (однонуклеотидные полиморфизмы) в последовательностях ДНК микробактерий туберкулеза. Цель заключается в нахождении таких участков генома, мутации в которых влияют на наличие либо отсутствие лекарственной устойчивости к определенному препарату. Для определения мутаций использовалось сравнение геномной последовательности с некоторой референсной последовательностью. Любое отличие символа в исходной последовательности и символа референсной последовательности считалось мутацией.

На вход поступает описание мутаций в последовательностях ДНК микробактерий туберкулеза. Для более наглядного представления введем матрицу X размера $n \times m$, где n – число наблюдаемых объектов, m – число различных позиций, в которых наблюдались мутации. Элемент матрицы x_{ij} равен 1, если у объекта под номером i наблюдалась мутация в -й по счету позиции среди тех, в которых наблюдалась мутация хотя бы одном среди имеющихся n объектов, $x_{ij} = 0$, если в этой же позиции не наблюдается мутация и $x_{ij} = -1$, если не удалось установить, была ли мутация в соответствующей позиции. Информация о чувствительности к определенному препарату закодирована в векторе y, где $y_i = 1$, если установлена лекарственная устойчивость -го объекта к выбранному препарату, $y_i = 0$, если у соответствующего объекта установлена лекарственная чувствительность к препарату и $y_i = -1$, если сведения отсутствуют.

Выходные данные представляют собой следующее:

1) Список мутаций, влияющих на чувствительность или устойчивость к препарату.

2) Модель, использующая описанный выше список мутаций, позволяющая для новых геномов микробактерий предсказывать, являются ли они устойчивыми или чувствительными к лекарственному препарату

1.2 Описание некоторых алгоритмов машинного обучения

Для решения подобной задачи часто используют методы машинного обучения. Существуют методы, позволяющие как отбирать значимые признаки (в

нашем случае, значимые мутации), так и строить модели для предсказания лекарственной устойчивости для новых объектов. Далее, рассмотрим некоторые широко известные алгоритмы машинного обучения.

1.2.1 Логистическая регрессия

Для описания логистической регрессии воспользуемся материалом из [8].

Логистическая регрессия применяется для предсказания вероятности возникновения некоторого события по значениям множества признаков. Для этого вводится так называемая зависимая переменная y, которая принимает лишь одно из двух значений — как правило, это числа 0 (событие не произошло) и 1 (событие произошло), и множество независимых переменных (также называемых признаками, предикторами или регрессорами) — вещественных $x_1, \dots x_n$, на основе значений которых требуется вычислить вероятность принятия того или иного значения зависимой переменной.

Делается предположение о том, что вероятность события $y = 1$ равняется:

$$P\{y = 1|x\} = f(z), \text{ где } z = \theta_1 x_1 + \cdots + \theta_n x_n \quad (1)$$

Здесь x и θ — векторы-столбцы значений независимых переменных $x_1, \dots, x_n$ и параметров (коэффициентов регрессии) — вещественных чисел $\theta_1, \dots, \theta_n$, соответственно, а $f(z)$ — так называемая *логистическая функция* (иногда также называемая сигмоидом или логит-функцией): $f(z) = \frac{1}{1+e^{-z}}$.

Для подбора параметров $\theta_1, \dots, \theta_n$ необходимо составить обучающую выборку, состоящую из наборов значений независимых переменных и соответствующих им значений зависимой переменной y . Формально, это множество пар $(x^{(1)}, y^{(1)}), \dots, (x^{(m)}, y^{(m)})$, где $x^i \in \mathbb{R}^n$ — вектор значений независимых переменных, а $y^{(i)} \in \{0,1\}$ — соответствующее им значение y . Каждая такая пара называется обучающим примером.

Обычно используется метод максимального правдоподобия, согласно которому выбираются параметры θ , максимизирующие значение функции правдоподобия на обучающей выборке.

Для улучшения обобщающей способности получающейся модели, то есть уменьшения эффекта переобучения, на практике часто рассматривается логистическая регрессия с регуляризацией.

Регуляризация заключается в том, что вектор параметров θ рассматривается как случайный вектор с некоторой заданной априорной плотностью распределения $p(\theta)$. Для обучения модели вместо метода наибольшего правдоподобия при этом используется метод максимизации апостериорной оценки.

В качестве априорного распределения часто выступает многомерное нормальное распределение с нулевым средним и матрицей ковариации, соответствующее априорному убеждению о том, что все коэффициенты регрессии должны быть небольшими числами, идеально — многие малозначимые коэффициенты должны быть нулями. Подставив плотность этого априорного распределения в формулу выше и прологарифмировав, получим следующую оптимизационную задачу:

$$\sum \log P\{y^i|x^i,\theta\} - \lambda\|\theta\|_2 \quad \to max \tag{2}$$

где $\lambda = const/\sigma^2$ — параметр регуляризации. Этот метод известен как L2-регуляризованная логистическая регрессия, так как в целевую функцию входит L2-норма вектора параметров для регуляризации.

Если вместо L2-нормы использовать L1-норму, что эквивалентно использованию распределения Лапласа, как априорного, вместо нормального, то получится другой распространённый вариант метода — L1-регуляризованная логистическая регрессия:

$$\sum \log P\{y^i|x^i,\theta\} - \lambda\|\theta\|_1 \to \max \tag{3}$$

1.2.2 Дерево принятия решений

Для описания дерева решений воспользуемся материалом из [9].

Дерево принятия решений (также может называться деревом классификации или регрессионным деревом) — средство поддержки принятия решений, использующееся в статистике и анализе данных для прогнозных моделей. Структура дерева представляет собой «листья» и «ветки». На ребрах («ветках») дерева решения записаны атрибуты, от которых зависит целевая функция, в «листьях» записаны значения целевой функции, а в остальных узлах — атрибуты, по которым различаются случаи. Чтобы классифицировать новый случай, надо спуститься по дереву до листа и выдать соответствующее значение. Подобные деревья решений широко используются в интеллектуальном анализе данных. Цель

состоит в том, чтобы создать модель, которая предсказывает значение целевой переменной на основе нескольких переменных на входе.

Каждый лист представляет собой значение целевой переменной, измененной в ходе движения от корня по листу. Каждый внутренний узел соответствует одной из входных переменных. Дерево может быть также «изучено» разделением исходных наборов переменных на подмножества, основанные на тестировании значений атрибутов. Это процесс, который повторяется на каждом из полученных подмножеств. Рекурсия завершается тогда, когда подмножество в узле имеет те же значения целевой переменной, таким образом, оно не добавляет ценности для предсказаний. Процесс, идущий «сверху вниз», индукция деревьев решений (TDIDT), является примером поглощающего «жадного» алгоритма, и на сегодняшний день является наиболее распространенной стратегией деревьев решений для данных, но это не единственная возможная стратегия. В интеллектуальном анализе данных, деревья решений могут быть использованы в качестве математических и вычислительных методов, чтобы помочь описать, классифицировать и обобщить набор данных, которые могут быть записаны следующим образом:

$$(x, Y) = (x_1, x_2, \dots x_k, Y) \quad (4)$$

Зависимая переменная Y является целевой переменной, которую необходимо проанализировать, классифицировать и обобщить. Вектор x состоит из входных переменных x_1, x_2, x_3 и т. д., которые используются для выполнения этой задачи.

1.2.3 Случайный лес

Для описания случайного леса воспользуемся материалом из [10].

Random forest (с англ. — «случайный лес») — алгоритм машинного обучения, предложенный Лео Брейманом и Адель Катлер [5], заключающийся в использовании комитета (ансамбля) решающих деревьев. Алгоритм сочетает в себе две основные идеи: метод бэггинга Бреймана, и метод случайных подпространств, предложенный Tin Kam Ho. Алгоритм применяется для задач классификации, регрессии и кластеризации.

Пусть обучающая выборка состоит из N примеров, размерность пространства признаков равна M, и задан параметр m (в задачах классификации обычно $m \approx \sqrt{M}$).

Все деревья комитета строятся независимо друг от друга по следующей процедуре:

1. Сгенерируем случайную подвыборку с повторением размером N из обучающей выборки.
2. Построим решающее дерево, классифицирующее примеры данной подвыборки, причём в ходе создания очередного узла дерева будем выбирать признак, на основе которого производится разбиение, не из всех M признаков, а лишь из m случайно выбранных. Выбор наилучшего из этих m признаков может осуществляться различными способами. В оригинальном коде Бреймана используется критерий Джини, применяющийся также в алгоритме построения решающих деревьев CART. В некоторых реализациях алгоритма вместо него используется критерий прироста информации.
3. Дерево строится до полного исчерпания подвыборки и не подвергается процедуре прунинга (англ. *pruning* — отсечение ветвей) (в отличие от решающих деревьев, построенных по таким алгоритмам, как CART или C4.5).

Классификация объектов проводится путём голосования: каждое дерево комитета относит классифицируемый объект к одному из классов, и побеждает класс, за который проголосовало наибольшее число деревьев.

Случайные леса, получаемые в результате применения техник, описанных ранее, могут быть естественным образом использованы для оценки важности переменных в задачах регрессии и классификации. Следующий способ такой оценки был описан Breiman.

Первые шаг в оценке важности переменной в тренировочном наборе — тренировка случайного леса на этом наборе. Во время процесса построения модели для каждого элемента тренировочного набора считается так называемая out-of-bag-ошибка. Затем для каждой сущности такая ошибка усредняется по всему случайному лесу.

Для того, чтобы оценить важность j-ого параметра после тренировки, значения j-ого параметра перемешиваются для всех записей тренировочного набора и out-of-bag-ошибка считается снова. Важность параметра оценивается путём усреднения по всем деревьям разности показателей out-of-bag-ошибок до и

после перемешивания значений. При этом значения таких ошибок нормализуются на стандартное отклонение.

Параметры выборки, которые дают бóльшие значения, считаются более важными для тренировочного набора. Метод имеет следующий потенциальный недостаток — для категориальных переменных с большим количеством значений метод склонен считать такие переменные более важными. Частичное перемешивание значений в этом случае может снижать влияние этого эффекта. Из групп коррелирующих параметров, важность которых оказывается одинаковой, выбираются меньшие по численности группы.

Достоинства:

- Способность эффективно обрабатывать данные с большим числом признаков и классов.
- Нечувствительность к масштабированию (и вообще к любым монотонным преобразованиям) значений признаков.
- Одинаково хорошо обрабатываются как непрерывные, так и дискретные признаки. Существуют методы построения деревьев по данным с пропущенными значениями признаков.
- Существует методы оценивания значимости отдельных признаков в модели.
- Внутренняя оценка способности модели к обобщению (тест out-of-bag).
- Высокая параллелизуемость и масштабируемость.

Недостатки:

- Алгоритм склонен к переобучению на некоторых задачах, особенно на зашумленных задачах.
- Большой размер получающихся моделей. Требуется $O(NK)$ памяти для хранения модели, где K — число деревьев.

1.2.4 Бустинг

Описание бустинга возьмем из [11].

Бустинг (англ. boosting — улучшение) — это процедура последовательного построения композиции алгоритмов машинного обучения, когда каждый следующий алгоритм стремится компенсировать недостатки композиции всех

предыдущих алгоритмов. Бустинг представляет собой жадный алгоритм построения композиции алгоритмов. Изначально понятие бустинга возникло в работах по вероятно почти корректному обучению в связи с вопросом: возможно ли, имея множество плохих (незначительно отличающихся от случайных) алгоритмов обучения, получить хороший.

В течение последних 10 лет бустинг остаётся одним из наиболее популярных методов машинного обучения, наряду с нейронными сетями и машинами опорных векторов. Основные причины — простота, универсальность, гибкость (возможность построения различных модификаций), и, главное, высокая обобщающая способность.

Бустинг над решающими деревьями считается одним из наиболее эффективных методов с точки зрения качества классификации. Во многих экспериментах наблюдалось практически неограниченное уменьшение частоты ошибок на независимой тестовой выборке по мере наращивания композиции. Более того, качество на тестовой выборке часто продолжало улучшаться даже после достижения безошибочного распознавания всей обучающей выборки. Это перевернуло существовавшие долгое время представления о том, что для повышения обобщающей способности необходимо ограничивать сложность алгоритмов. На примере бустинга стало понятно, что хорошим качеством могут обладать сколь угодно сложные композиции, если их правильно настраивать.

Впоследствии феномен бустинга получил теоретическое обоснование. Оказалось, что взвешенное голосование не увеличивает эффективную сложность алгоритма, а лишь сглаживает ответы базовых алгоритмов. Количественные оценки обобщающей способности бустинга формулируются в терминах отступа. Эффективность бустинга объясняется тем, что по мере добавления базовых алгоритмов увеличиваются отступы обучающих объектов. Причём бустинг продолжает раздвигать классы даже после достижения безошибочной классификации обучающей выборки.

К сожалению, теоретические оценки обобщающей способности дают лишь качественное обоснование феномену бустинга. Хотя они существенно точнее более общих оценок Вапника-Червоненкиса, всё же они сильно завышены, и требуемая длина обучающей выборки оценивается величиной порядка $10^4 - 10^6$. Более основательные эксперименты показали, что иногда бустинг всё же переобучается.

1.3 Основные методы отбора признаков

Отбор признаков является одним из важных этапов решения задач машинного обучения. Поэтому, придумано достаточно много методов отбора признаков. Далее, приведем некоторые основные подходы, использующиеся при отборе признаков.

1.3.1 Подход, основанный на статистических методах

Суть таких методов заключается в статистической оценке значимости признаков. Как правило, признаки рассматриваются по отдельности и для каждого признака независимо строится мера его значимости (например, с помощью теста хи-квадрат независимости признака и скрытой переменной). Далее, выбираются признаки, мера значимости которых выше некоторого порога, который зачастую является параметром алгоритма. Преимуществом такого подхода является его относительная простота и зачастую небольшая вычислительная сложность. Однако основным недостатком является то, что мы перебираем каждый признак по отдельности и не учитываем признаки в совокупности.

1.3.2 Определение значимости признаков исходя из параметров обученной модели

Суть таких методов заключается в том, что параметры обученных моделей машинного обучения могут быть использованы для определения значимости признаков. Например, в логистической регрессии обученный вектор параметров è уже даёт веса признаков по отношению к скрытой переменной. Например, высокое положительное или низкое отрицательное значение $è_i$ говорит о том, что значение признака x_i сильно влияет на предсказание модели, а, следовательно, значимо для неё. В методах, основанных на деревьях решений, ясно, что если признак, встречающийся в дереве, более значим, чем признак, в дереве не встречающийся. На основе этой информации, а также информации о том, как именно разделяет выборку значение признака, можно оценить его значимость для модели. В отличие от предыдущего подхода, признаки исследуются уже в совокупности (ведь именно совокупность признаков влияет на модель). Однако недостатком является более сложная интерпретируемость результатов, а также их зависимость от конкретной модели.

1.3.3 Методы отбора признаков, встраиваемые в модели

Это методы, которые устроены таким образом, что незначимые признаки исключаются из модели и реально ей не используются. Например, таким свойством обладает регуляризация L1 в линейных моделях. Плюсом является

удобство, ведь модель “самостоятельно определяет”, какие признаки ей нужны, без отдельной стадии обработки данных. Недостатком модели, как и в предыдущем случае, является зависимость результатов от используемого алгоритма машинного обучения. Также, не все модели позволяют встраивание в них отбора признаков.

1.3.4 Определение множества значимых признаков на основе качества модели, использующей множество признаков

Суть таких методов заключается в том, что мы можем перебирать различные подмножества признаков (конкретный способ перебора и определяет алгоритм), и для каждого перебираемого подмножества тестировать модель, обученную на выбранном подмножестве признаков (это можно делать посредством скользящего контроля). Метрикой качества при этом может выступать функция, зависящая от качества модели и отобранных признаков (в более простом случае, количества признаков). В качестве результата выступает множество признаков, на котором значение метрики качества максимально. Преимуществом метода является применимость к любым моделям, а также тот факт, что правильная метрика качества отражает результат, который мы в итоге ожидаем от модели (высокая точность и, например, небольшое число признаков). Недостатком является вычислительная сложность (необходимо много раз обучать модель), а также невозможность перебирать все варианты (так как их экспоненциальное число от количества признаков), что вынуждает придумывать нетривиальные алгоритмы перебора множества признаков.

1.4 Некоторые известные факты из теории графов

Для того, чтобы лучше понять основные используемые далее факты из теории графических моделей, вспомним несколько важных фактов их теории графов.

1.4.1 Потоки в сетях

Рассмотрим неориентированный граф с двумя выделенными вершинами (стоком t и истоком s). Пусть с каждым ребром $(u, v) \in E$ ассоциировано некоторое неотрицательное число $c(u, v) \geq 0$ - пропускная способность. Назовем потоком неотрицательную функцию $f(u, v)$, определенную на ребрах графа, такую что $f(u, v) \leq c(u, v)$, при этом $\sum_{v:(u,v)\in E} f(u, v) - \sum_{v:(v.u)\in E} f(v, u) = 0, \forall u \notin \{s, t\}$. Первое условие ограничивает поток через ребро его пропускной

способностью, а второе гарантирует отсутствие источников и стоков вне выделенной пары вершин. Задача поиска максимального потока состоит в максимизации величины $M(f) = \sum_{v:(s,v)\in E} f(s,v) \to max$ по всем допустимым потокам.

1.4.2 Разрезы графа

st -разрезом графа называется разбиение вершин графа на два непересекающихся множества S и T, такие что $s \in S, t \in T$. Величиной разреза называется сумма пропускных способностей всех ребер, один конец которых находится в множестве S, а другой – в множестве T.

1.4.3 Теорема Форда-Фалкерсона

Известная теорема Форда-Фалкерсона гласит, что величина максимального поток в сети равна величине ее минимального разреза. Существует эффективный (полиномиальный) алгоритм решения задачи минимального разреза в графе.

1.5 Графическая вероятностная модель

Далее, опишем основные понятия, связанные с аппаратом графических моделей, воспользовавшись [12].

1.5.1 Понятие графической вероятностной модели

Графическая вероятностная модель — это вероятностная модель, в которой в виде графа представлены зависимости между случайными величинами. Вершины графа соответствуют случайным переменным, а рёбра — непосредственным вероятностным взаимосвязям между случайными величинами. Графические модели широко используются в теории вероятностей, статистике (особенно в Байесовской статистике), а также в машинном обучении.

1.5.2 Байесовская сеть

Байесовская сеть (или *байесова сеть, байесовская сеть доверия*, англ. *Bayesian network, belief network*) — графическая вероятностная модель, представляющая собой множество переменных и их вероятностных зависимостей по Байесу. Например, байесовская сеть может быть использована для вычисления вероятности того, чем болен пациент по наличию или отсутствию ряда симптомов, основываясь на данных о зависимости между симптомами и болезнями. Математический аппарат байесовых сетей создан американским учёным Джудой Перлом, лауреатом Премии Тьюринга (2011).

Формально, байесовская сеть — это направленный ациклический граф, каждой вершине которого соответствует случайная переменная, а дуги графа кодируют отношения условной независимости между этими переменными. Вершины могут представлять переменные любых типов, быть взвешенными параметрами, скрытыми переменными или гипотезами. Существуют эффективные методы, которые используются для вычислений и обучения байесовских сетей. Если переменные байесовской сети являются дискретными случайными величинами, то такая сеть называется дискретной байесовской сетью. Байесовские сети, которые моделируют последовательности переменных, называют динамическими байесовскими сетями. Байесовские сети, в которых могут присутствовать как дискретные переменные, так и непрерывные, называются гибридными байесовскими сетями. Байесовская сеть, в которой дуги помимо отношений условной независимости кодируют также отношения причинности, называют причинно-следственными байесовыми сетями (англ. *causal bayesian networks*)).

Если из вершины A выходит дуга в вершину B, то A называют *родителем* B, а B называют *потомком* A. Если из вершины A существует ориентированный путь в вершину B, то A называется *предком* B, а B называется *потомком* A. Множество вершин-родителей вершины V_i обозначим как $parents(V_i) = PA_i$.

Направленный ациклический граф G называется *байесовской сетью* для вероятностного распределения $P(v)$, заданного над множеством случайных переменных V, если каждой вершине графа поставлена в соответствие случайная переменная из V, а дуги в графе удовлетворяют условию (марковское условие): любая переменная V_i из V должна быть условно независима от всех вершин, не являющихся её потомками, если заданы (получили означивание, обусловлены) все её прямые родители PA_i в графе G, то есть $\forall V_i \in V$ справедливо: $P(v_i \mid pa_i, s) = P(v_i \mid pa_i)$, где v_i — значение V_i; s — конфигурация S; S — множество всех вершин, не являющихся потомками V_i; pa_i — конфигурация PA_i.

Тогда полное совместное распределение значений в вершинах можно удобно записать в виде декомпозиции (произведения) локальных распределений:

$$P(V_1, \dots, V_n) = \prod_{i}^{n} P(V_i | parents(V_i))$$

Если у вершины V_i нет предков, то её локальное распределение вероятностей называют *безусловным*, иначе *условным*. Если вершина — случайная переменная получила означивание (например, в результате наблюдения), то такое означивание называют *свидетельством* (англ. *evidence*).

1.5.3 Марковская сеть

Марковская сеть, марковское случайное поле, или неориентированная графическая модель — это графическая модель, в которой множество случайных величин обладает Марковским свойством, описанным неориентированным графом. Марковская сеть отличается от другой графической модели, Байесовской сети, представлением зависимостей между случайными величинами. Она может выразить некоторые зависимости, которые не может выразить Байесовская сеть (например, циклические зависимости); с другой стороны, она не может выразить некоторые другие. Прототипом Марковской сети была модель Изинга магничивания материала в статистической физике: Марковская сеть была представлена как обобщение этой модели.

Неориентированный граф $G = (V, E)$, множество случайных величин $(X_v), v \in V$образуют марковское случайное поле по отношению к G, если они удовлетворяют следующим эквивалентным марковским свойствам:

- Свойство пар: Любые две несмежные переменные условно независимы с учетом всех других переменных.
- Локальное свойство: переменная условно независима от всех других величин, с учетом своих соседей.
- Глобальное свойство: Любые два подмножества переменных условно независимы с учетом разделяющего подмножества.

Другими словами, граф G считается марковским случайным полем по отношению к совместному распределению вероятностей $P\,(X = \mathrm{x})$ на множестве случайных величин X тогда и только тогда, когда разделение графа G подразумевает условную независимость: если два узла и разделены в G после удаления из G множества узлов Z, то $P\,(X = \mathrm{x})$ должна утверждать, что x_iи x_j условно независимы с учетом случайных величин, соответствующих Z. Если это условие выполнено, то говорят, что G является независимой картой (independency map) (или И-картой (I-map)) распределения вероятностей.

Многие определения требуют чтобы G было минимальной И-картой, то есть И-картой, при удалении из которой одного ребра она перестает быть И-картой. Это разумно требовать, поскольку это приводит к наиболее компактному представлению, которое включает как можно меньше зависимостей; отметим, что полный граф это тривиальная И-карта. В случае, когда G не только И-карта (то есть не представляет независимости, которые не указаны в $P\,(X = \mathrm{x})$), но и не представляет зависимости, которые не указаны в $P\,(X = \mathrm{x})$, G называется

совершенной картой (perfect map) $P\,(X = \text{x})$. Она представляет набор независимостей указанных $P\,(X = \text{x})$.

Марковские случайные сети получили широкое распространение в задачах обработки изображения (например, в сегментации изображений), в задачах обработки естественного языка (например, при выделении частей речи, определения именованных сущностей).

1.5.4 Энергия марковской сети

Обозначим *энергией* конфигурации x величину $-\ln P(X = x)$. Из определения видно, что чем вероятней конфигурация x, тем меньше её энергия. Задачей минимизации энергии называется задача поиска такой конфигурация, энергия которой минимальна.

Известно, что энергию всегда можно представить в следующем виде:

$$E(x) = \sum_{X_i \in V} \theta(x_i) + \sum_{c \in C} \theta(x_c), \tag{5}$$

где C – множество клик графа. Слагаемые правой части описанной формулы называются *потенциалами.*

Потенциал будем считать *унарным, если* соответствующая ему клика состоит из единственной вершины и *парным, если* соответствующая клика состоит из двух вершин.

Далее, рассмотрим один важный случай, в котором задача минимизации энергии решается точно и за полиномиальное время.

Энергии, состоящие только из унарных и парных потенциалов (то есть, если ее можно представить в виде суммы унарных и парных потенциалов) назовём парно-сепарабельными. Парно-сепарабельные энергии будем записывать следующим способом:

$$E(x) = \sum_{i \in V} \theta(x_i) + \sum_{\{i,j\} \in E} \theta_{ij}(x_i, x_j) \tag{6}$$

.Далее, приведем описание важного частного случая парно-сепарабельной энергии, взятое из [3].

Рассмотрим задачу минимизации парно-сепарабельной энергии, заданной на произвольном графе $G = (V, E)$. Пусть все переменные бинарны: $P = \{0, 1\}$. Функции, отображающие булев куб на множество действительных чисел, часто называют псевдо-булевыми. Известно, что если не вводить никаких

дополнительных ограничений, то задача минимизации парно-сепарабельной псевдо-булевой функции является N -трудной. Однако если все парные потенциалы такой функции удовлетворяют условию субмодулярности, то задачу можно решить за полиномиальное время при помощи сведения к задачам построения максимального потока и минимального разреза в графе. Данный результат был известен ещё в середине 60-х, но был переоткрыт в начале нулевых В. Колмогоровым и др. С тех пор алгоритмы минимизации энергии, основанные на использовании алгоритмов построения разрезов графов (graph cuts) активно используются в задачах компьютерного зрения и машинного обучения.

Далее, введем понятие субмодулярности, а также приведем доказательство разрешимости задачи минимизации энергии за полиномиальное время. Данное доказательство полезно своей конструктивностью.

Функция бинарных переменных $E(x)$ называется *субмодулярной*, если для любых двух разметок x_1 и x_2 выполнено условие $E(x^1 \vee x^2) + E(x^1 \wedge x^2) \leq \leq E(x^1) + E(x^2)$, где операции «$\vee$» и «$\wedge$» – поэлементная дизъюнкция (максимум) и конъюнкция (минимум), соответственно.

*Утверждение***.** Парно-сепарабельная функция бинарных переменных $E(x)$ является субмодулярной, тогда и только тогда, когда для каждого парного потенциала выполнено следующее условие: $\theta_{ij}(0,0) + \theta_{ij}(1,1) \leq \theta_{ij}(0,1) + +\theta_{ij}(1,0)$.

Описанное выше утверждение можно воспринимать так: энергия парных потенциалов меньше в тех случаях, когда конфигурации соседних вершин совпадают. При решении некоторых задач данное условие субмодулярности выполняется естественным образом. Например, при решении задачи отделения объекта от фона. Пусть каждый пиксель изображения представлен вершиной в графе V, а каждая пара соседних по стороне пикселей образует ребро между соответствующими вершинами графа. Интуитивно понятно, что пиксели объекта будут располагаться в одной или нескольких связных областях изображения. Это как раз и означает выполнение условия субмодулярности. Далее, мы увидим, что данное условие выполняется и в сети релевантных признаков. Там потенциалы выставляются так, чтобы коррелированные признаки были с большей вероятностью либо оба релевантными, либо оба нерелевантными.

Рассмотрим парно-сепарабельную энергию бинарных переменных, в которой потенциалы удовлетворяют следующим ограничениям:

• унарные потенциалы неотрицательны: $\forall i \in V\ \theta_i(0) > 0, \theta_i(1) > 0$;

• парные потенциалы неотрицательны и равны 0 при равенстве связанных переменных: $\forall\{i,j\} \in E\ \theta_{ij}(0,0) = \theta_{ij}(1,1) = 0, \theta_{ij}(0,1) > 0, \theta_{ij}(1,0) > 0$.

В этом случае энергию можно записать в следующем виде:

$$E(x) = \sum_{i \in V} \big(x_i \theta_i(1) + (1 - x_i)\theta_i(0)\big) + \\ + \sum_{\{i,j\} \in E} \Big(x_i\big(1 - x_j\big)\theta_{ij}(1,0) + x_j(1 - x_i)\theta_{ij}(0,1)\Big) + \theta_0 \quad (7)$$

По энергии построим ориентированный граф $G^- = (V^-, E^-)$, st-разрез которого будет соответствовать присвоению значений переменным x.

• Множество вершин $V^- = V \cup \{s,t\}$, где s и t – две дополнительные вершины: исток и сток, соответственно.

• Множество дуг E^- строится следующим образом: каждому ребру $\{i,j\} \in E$ ставим в соответствие две дуги (i,j) и (j,i) (нетерминальные дуги), каждой вершине $i \in V$ ставим в соответствие дуги (s,i) и (i,t) (терминальные дуги).

• st-разрезом графа будем считать разбиение всех вершин множества V^- на два непересекающихся множества S и T, $S \cap T = \emptyset$, $S \cup T = V^-$, такое что $s \in S$ и $t \in T$. Будем считать, что множество истока S соответствует значению переменных 0, а множество стока T – значению 1.

• Веса терминальных дуг: $c(s,i) = \theta_i(1), c(i,t) = \theta_i(0)$, где $i \in V$.

• Веса нетерминальных дуг: $c(i,j) = \theta_{ij}(0,1), c(j,i) = \theta_{ij}(1,0)$, где $\{i,j\} \in E$, и, по договоренности, $i < j$.

• Присваивание значений переменных по построенному st-разрезу строится следующим образом: $i \in S \Rightarrow x_i = 0, i \in T \Rightarrow x_i = 1$. Величина разреза совпадает со значением энергии при таких значениях переменных x. Таким образом, задача минимизации энергии эквивалента задаче поиска минимального s-разреза в графе G^-.

Рассмотрим, какие ещё энергии можно минимизировать при помощи разрезов графов. На энергию $E(x)$ можно смотреть как на функцию. У одной и той же функции может быть несколько различных записей вида. Назовем преобразование записи энергии $E(x)$, не меняющее энергию, как функцию, репараметризацией. Рассмотрим несколько видов репараметризаций:

• Вычитание константы из унарного потенциала: $\theta_i(0) := \theta_i(0) -$

$- u, \theta_i(1) := \theta_i(1) - u, \theta_0 := \theta_0 + u.$ Здесь $i \in V, u \in \mathbb{R}$.

• Изменение парных потенциалов: $\theta_{ij}(p,0) := \theta_{ij}(p,0) - u, \theta_{ij}(p,1) :=$

$:= \theta_{ij}(p,1) - u, \theta_i(p) := \theta_i(p) + u$. Аналогично $\theta_{ij}(0,p) := \theta_{ij}(0,p) -$

$- u, \theta_{ij}(1,p) := \theta_{ij}(1,p) - u, \theta_j(p) := \theta_j(p) + u$. Здесь $\{i,j\} \in E$, $u \in R, p \in \{0,1\}$. Легко показать, что описанные преобразования не меняют энергию $E(x)$ как функцию. Рассмотрим, какие парно-сепарабельные энергии общего вида при помощи описанных преобразований-репараметризаций можно свести к виду, для которого мы уже умеем решать задачу минимизации энергии. Заметим, что применяя репараметризацию унарных потенциалов с $u = \min(\theta_i(0), \theta_i(1))$ унарные потенциалы всегда можно сделать неотрицательными. Таким образом, можно снять все ограничения на унарные потенциалы энергии. Рассмотрим, что можно сделать при помощи репараметризаций парных потенциалов. Пусть потенциал ребра $\{i,j\} \in E$ принимает следующие значения: $\theta_{ij}(0,0) = a, \theta_{ij}(1,1) = b, \theta_{ij}(0,1) =$

$= c, \theta_{ij}(1,0) = d$. Применим следующую последовательность преобразований:

1) $\theta_{ij}(0,0) := \theta_{ij}(0,0) - a, \theta_{ij}(0,1) := \theta_{ij}(0,1) - a, \theta_i(0) := \theta_i(0) + a;$
2) $\theta_{ij}(0,1) := \theta_{ij}(0,1) - c + a, \theta_{ij}(1,1) := \theta_{ij}(1,1) - c + a, \theta_j(1) :=$
   $:= \theta_j(1) + c - a;$
3) $\theta_{ij}(1,1) := \theta_{ij}(1,1) - b + c - a, \theta_{ij}(1,0) := \theta_{ij}(1,0) - b + c -$
   $-a, \theta_i(1) := \theta_j(1) + b - c + a.$

Можно убедиться, что данные преобразования приводят к следующим значениям парного потенциала θ_{ij}: $\theta_{ij}(0,0) = \theta_{ij}(1,1) = \theta_{ij}(0,1) = 0, \theta_{ij}(1,0) = d + c - a - b$. Если после этого при помощи описанных ранее действий сделать все унарные потенциалы неотрицательными, то энергия будет иметь вид (1), тогда и только тогда, когда выполнено условие $d + c - a - b > 0$. Заметим, что это условие в точности

соответствует условию субмодулярности парно-сепарабельной энергии бинарных переменных.

Таким образом, задачу минимизации энергии парно-сепарабельной субмодулярной Марковской сети можно эффективно решить при помощи алгоритма нахождения максимального потока сети.

1.6 Сеть релевантных признаков

Сеть релевантных признаков предложена и описана в [1]. Это достаточно интересная модель, позволяющая применять аппарат графических моделей к задаче отбора признаков. Рассмотрим ее более подробно.

Предположим, мы решаем задачу обучения с учителем, имея n объектов и d признаков ($d \gg n$) . Сеть релевантных признаков - это бинарная парно-сепарабельная субмодулярная марковская сеть, описывающая случайный вектор $X = (X_1, \dots, X_d) \in \{0,1\}^d$, распределение которого описывается графом $G = (V, E)$ и потенциалами его вершин и ребер. Признак i описывается вершиной v_i. $X_i = 1$ означает, что признак i является релевантным, и $X_i = 0$ в противном случае. Корреляция между признаками i и j определяется соответствующим ребром (v_i, v_j) в графе G. Потенциал è$_i$ вершины v_i определяется уверенностью в том, что признак i является значимым, а парный потенциал è$_{ij}$ определяется корреляцией признаков i и j, а также зависит от того, принимают ли величины x_i и x_j одинаковые значения.

Сеть релевантных признаков может быть полезна для того, чтобы скорректировать значимость признаков, полученная другим методом. Например, если признак не очень значим, при этом сильно коррелирует с другими значимыми признаками, наверняка стоит повысить степень его значимости и наоборот, если признак признан значимым, при этом признаки, с которыми он коррелирует, менее значимы, стоит понизить значимость признака (возможно, значимость признака является случайностью).

Рассмотрим, как в сети релевантных признаков задаются одиночные и парные потенциалы. Пусть нам кроме значения признаков и скрытой переменной дано значение параметра ë ≥ 0, а также p_i - значимость признака i. Такая функция может быть построена с помощью одного из подходов, описанных в разделе 1.3. Тогда, одиночные потенциалы è$_i$ и парные потенциалы è$_{ij}$ определены следующим образом:

- $\theta_i = |X_i - p_i|, i \in [1, d]$
- $\theta_{ij} = \lambda|r_{ij}|[X_i \neq X_j]$ где r_{ij} - корреляция между признаками i и j

Нетрудно видеть, что при такой конфигурации сеть действительно будет бинарной, парно-сепарабельной и субмодулярной, что допускает точное решение с помощью алгоритма поиска максимального потока в сети.

Также, можно модифицировать одиночные потенциалы, введя новый параметр:

$$\theta_i = |2\alpha X_i - p_i|, i \in [1, d]$$

Заметим, что при $\alpha = 0.5$ значение одиночного потенциала аналогично предыдущему случаю. Параметр предназначен для регулирования того, какие признаки считаются важными. Например, в исходном варианте, релевантными считаются признаки, вероятность значимости которых не меньше 0.5. Мы же можем воспринимать релевантным признак, вероятность релевантности которого не меньше 0.9, что достигается использованием $\alpha = 0.9$.

1.7 Оценивание качества алгоритма методом скользящего контроля

Для оценки качества алгоритмов машинного обучения наиболее часто используются методы скользящего контроля. Остановимся на них поподробнее. Последующие материалы использованы из описания скользящего контроля в статье [13].

1.7.1 Общее описание

Скользящий контроль или кросс-валидация (cross-validation, CV) — процедура эмпирического оценивания обобщающей способности алгоритмов.

Фиксируется некоторое множество разбиений исходной выборки на две подвыборки: *обучающую* и *контрольную*. Для каждого разбиения выполняется настройка алгоритма по обучающей подвыборке, затем оценивается его средняя ошибка на объектах контрольной подвыборки. *Оценкой скользящего контроля* называется средняя по всем разбиениям величина ошибки на контрольных подвыборках.

Если выборка независима, то средняя ошибка *скользящего контроля* даёт несмещённую оценку вероятности ошибки. Это выгодно отличает её от средней ошибки на обучающей выборке, которая может оказаться смещённой

(оптимистически заниженной) оценкой вероятности ошибки, что связано с явлением переобучения.

Скользящий контроль является стандартной методикой тестирования и сравнения алгоритмов классификации, регрессии и прогнозирования.

Рассматривается задача обучения с учителем.

Пусть X — множество описаний объектов, Y — множество допустимых ответов.

Задана конечная выборка прецедентов $X^L = (x_i, y_i)_{i=1}^L \subset X \times Y$.

Задан алгоритм обучения — отображение ì, которое произвольной конечной выборке прецедентов X^m ставит в соответствие функцию (алгоритм) $a: X \to Y$.

Качество алгоритма a оценивается по произвольной выборке прецедентов X^m с помощью *функционала качества* $Q(a, X^m)$. Для процедуры скользящего контроля не важно, как именно вычисляется этот функционал. Как правило, он аддитивен по объектам выборки: $Q(a, X^m) = \frac{1}{m}\sum_{x_i \in X^m} L(a(x_i), y_i)$

где $L(a(x_i), y_i)$ — неотрицательная функция потерь, возвращающая величину ошибки ответа алгоритма $a(x_i)$ при правильном ответе y_i.

1.7.2 Процедура скользящего контроля

Выборка X^L разбивается N различными способами на две непересекающиеся подвыборки: $X^L = X_n^m \cup X_n^k$, где X_n^m — обучающая подвыборка длины m, X_n^k — контрольная подвыборка длины $k = L - m$, $n = 1, \dots, N$ — номер разбиения.

Для каждого разбиения n строится алгоритм $a_n = a(X_n^m)$ и вычисляется значение функционала качества $Q_n = Q(a_n, X_n^k)$. Среднее арифметическое значений Q по всем разбиениям называется *оценкой скользящего контроля*:

$$CV(\text{ì}, X^L) = \frac{1}{N}\sum_{n=1}^{N} Q(a(X_n^m), X_n^k)$$

Различные варианты скользящего контроля отличаются видами функционала качества и способами разбиения выборки.

1.7.3 Доверительное оценивание

Кроме среднего значения качества на контроле строят также доверительные интервалы.

Непараметрическая оценка доверительного интервала. Строится вариационный ряд значений $Q_n = Q(a_n, X_n^k), n = 1, \dots, N$:

$$Q^{(1)} \leq Q^{(2)} \leq \cdots \leq Q^{(n)}$$

Утверждение 1. Если разбиения осуществлялись случайно, независимо и равновероятно, то значение случайной величины $Q(a(X^m), X^k)$ не превосходит $Q^{(N-t+1)}$ с вероятностью $p = \frac{t}{N+1}$.

Следствие 1. Значение случайной величины $Q(a(X^m), X^k)$ не превосходит $Q^{(N)}$ с вероятностью $p = 1/(N+1)$. В частности, для получения верхней оценки с надёжностью 95% достаточно взять $N = 20$ разбиений.

Утверждение 2. Если разбиения осуществлялись случайно, независимо и равновероятно, то с вероятностью $p = 2t/(N+1)$ значение случайной величины $Q(a(X^m), X^k)$ не выходит за границы доверительного интервала $\left[Q^{(t)}, Q^{N-t+1}\right]$.

Следствие 2. Значение случайной величины $Q(a(X^m), X^k)$ не выходит за границы вариационного ряда $\left[Q^{(1)}, Q^{(N)}\right]$ с вероятностью $p = \frac{2}{N+1}$.

В частности, для получения двусторонней оценки с надёжностью 95% достаточно взять $N = 40$ разбиений.

Параметрические оценки доверительного интервала основаны на априорном предположении о виде распределения случайной величины $Q(a(X^m), X^k)$. Если априорные предположения не выполняются, доверительный интервал может оказаться сильно смещённым. В частности, если предположения о нормальности распределения не выполнены, то нельзя пользоваться стандартным «правилом двух сигм» или «трёх сигм». Джон Лангфорд в своей диссертации указывает на распространённую ошибку, когда правило двух сигм применяется к функционалу частоты ошибок, имеющему на самом деле биномиальное распределение. Однако биномиальным распределением в общем случае тоже пользоваться нельзя, поскольку в результате обучения по случайным подвыборкам X^m вероятность ошибки алгоритма $a(X^m)$ оказывается случайной величиной. Следовательно, случайная величина $Q(a(X^m), X^k)$ описывается не биномиальным распределением, а (неизвестной) смесью биномиальных распределений. Аппроксимация смеси биномиальным распределением может приводить к

ошибочному сужению доверительного интервала. Приведённые выше непараметрические оценки лишены этого недостатка.

1.7.4 Стратификация

Стратификация выборки — это способ уменьшить разброс (дисперсию) оценок скользящего контроля, в результате чего получаются более узкие доверительные интервалы и более точные (tight) верхние оценки.

Стратификация заключается в том, чтобы заранее поделить выборку на части (страты), и при разбиении на обучение длины m и контроль длины k гарантировать, что каждая страта будет поделена между обучением и контролем в той же пропорции m: k.

Стратификация классов в задачах классификации означает, что каждый класс делится между обучением и контролем в пропорции m: k.

Стратификация по вещественному признаку. Объекты выборки сортируются согласно некоторому критерию, например, по возрастанию одного из признаков. Затем выборка разбивается на k последовательных страт одинаковой (с точностью до 1) длины. При формировании контрольных выборок из каждой страты выбирается по одному объекту, либо с заданным порядковым номером внутри страты, либо случайным образом.

1.7.5 Разновидности скользящего контроля

Возможны различные варианты скользящего контроля, отличающиеся способами разбиения выборки.

Полный скользящий контроль (complete CV).

Оценка скользящего контроля строится по всем $N = C_L^k$ разбиениям. В зависимости от k (длины обучающей выборки) различают:

- Частный случай при $k = 1$ — контроль по отдельным объектам (leave-one-out CV);

Было показано, что контроль по отдельным объектом является асимптотически оптимальным при некоторых условиях..

- Общий случай при $k > 2$. Здесь число разбиений $N = C_L^k$ становится слишком большим даже при сравнительно малых значениях k, что затрудняет практическое применение данного метода. Для этого случая *полный скользящий контроль* используется либо в теоретических исследованиях, либо в тех редких

ситуациях, когда для него удаётся вывести эффективную вычислительную формулу. Например, такая формула известна для метода k ближайших соседей, что позволяет эффективно выбирать параметр k. На практике чаще применяются другие разновидности *скользящего контроля*.

Случайные разбиения

Разбиения $n = 1, \dots, N$ выбираются случайно, независимо и равновероятно из множества всех C_L^k разбиений. Именно для этого случая справедливы приведённые выше оценки доверительных интервалов. На практике эти оценки, как правило, без изменений переносится и на другие способы разбиения выборки.

Контроль на отложенных данных (hold-out CV)

Оценка скользящего контроля строится по одному случайному разбиению, $N = 1$.

Этот способ имеет существенные недостатки:

1. Приходится слишком много объектов оставлять в контрольной подвыборке. Уменьшение длины обучающей подвыборки приводит к смещённой (пессимистически завышенной) оценке вероятности ошибки.

2. Оценка существенно зависит от разбиения, тогда как желательно, чтобы она характеризовала только алгоритм обучения.

3. Оценка имеет высокую дисперсию, которая может быть уменьшена путём усреднения по разбиениям.

Следует различать скользящий контроль по отложенным данным и контроль по тестовой выборке. Если во втором случае оценивается вероятность ошибки для классификатора, построенного по обучающей подвыборке, то в первом случае - для классификатора, построенного по полной выборке (то есть доля ошибок вычисляется не для того классификатора, который выдается в качестве результата решения задачи).

Контроль по отдельным объектам (leave-one-out CV)

Является частным случаем полного скользящего контроля при $k = 1$, соотвественно, $N = L$. Это, пожалуй, самый распространённый вариант скользящего контроля.

Преимущества LOO в том, что каждый объект ровно один раз участвует в контроле, а длина обучающих подвыборок лишь на единицу меньше длины полной выборки.

Недостатком LOO является большая ресурсоёмкость, так как обучаться приходится L раз. Некоторые методы обучения позволяют достаточно быстро перенастраивать внутренние параметры алгоритма при замене одного обучающего объекта другим. В этих случаях вычисление LOO удаётся заметно ускорить.

Контроль по $\boldsymbol{q}$ блокам ($\boldsymbol{q}$-fold CV)

Выборка случайным образом разбивается на q непересекающихся блоков одинаковой (или почти одинаковой) длины $k_1, \dots k_q$: $X^L = X_1^{k_1} \cup \dots \cup X_q^{k_q}$

$k_1 + \dots + k_q = L$. Каждый блок по очереди становится контрольной подвыборкой, при этом обучение производится по остальным $q - 1$ блокам. Критерий определяется как средняя ошибка на контрольной подвыборке: $CV(a, X^L) = \frac{1}{q}\sum_{n=1}^{q} Q(a(X^L \backslash X_n^{k_n}), X_n^{k_n})$

Это компромисс между LOO, hold-out и случайными разбиениями. С одной стороны, обучение производится только q раз вместо L. С другой стороны, длина обучающих подвыборок, равная $L(q-1)/N$ с точностью до округления, не сильно отличается от длины полной выборки L. Обычно выборку разбивают случайным образом на 10 или 20 блоков.

Контроль по $\boldsymbol{r} \times \boldsymbol{q}$ блокам ($\boldsymbol{r} \times \boldsymbol{q}$-fold CV)

Контроль по q блокам (q-fold CV) повторяется r раз. Каждый раз выборка случайным образом разбивается на q непересекающихся блоков. Этот способ наследует все преимущества q-fold CV, при этом появляется дополнительная возможность увеличивать число разбиений.

Данный вариант скользящего контроля, со стратификацией классов, является стандартной методикой тестирования и сравнения алгоритмов классификации. В частности, он применяется в системах WEKA и «Полигон алгоритмов».

1.8 Метрики качества в задаче классификации

Так как в результате мы будем строить модель, позволяющую для нового объекта предсказывать устойчивость к препарату, то есть, решать задачу классификации, а при тестировании алгоритма необходимо выбрать метрику

качества, рассмотрим основные метрики, используемые в задаче классификации. Рассмотренное ниже описание составлено с помощью статьи [14].

1.8.1 Правильность

Наверно, самая простая метрика, которая определяется как доля объектов, которые алгоритм классифицировал верно, к числу всех объектов.

$$Accuracy = \frac{P}{N}$$

Где P – число верно классифицированных объектов, N – общее число объектов.

1.8.2 Точность и полнота

Точность (precision) и полнота (recall) являются метриками которые используются при оценке большей части алгоритмов извлечения информации. Иногда они используются сами по себе, иногда в качестве базиса для производных метрик, таких как -мера или $R - Precision$. Суть точности и полноты очень проста.

Точность системы в пределах класса – это доля документов действительно принадлежащих данному классу относительно всех документов которые система отнесла к этому классу. Полнота системы – это доля найденных классификатором документов принадлежащих классу относительно всех документов этого класса в тестовой выборке.

Эти значения легко рассчитать на основании таблицы контингентности, которая составляется для каждого класса отдельно. Таблица контингентности представлена таблицей 1.

		Экспертная оценка	
		Положительная	Отрицательная
Оценка система	Положительная	TP	FP
	Отрицательная	FN	TN

Таблица 1 - таблица контингентности

В таблице содержится информация сколько раз система приняла верное и сколько раз неверное решение по документам заданного класса. А именно:

- TP — истино-положительное решение;
- TN — истино-отрицательное решение;
- FP — ложно-положительное решение;
- FN — ложно-отрицательное решение.

Тогда, точность и полнота определяются следующим образом:

$$Precision = \frac{TP}{TP + FP},$$
$$Recall = \frac{TP}{TP + FN}$$

1.8.3 Матрица неточностей

На практике значения точности и полноты гораздо более удобней рассчитывать с использованием матрицы неточностей (confusion matrix). В случае если количество классов относительно невелико (не более 100-150 классов), этот подход позволяет довольно наглядно представить результаты работы классификатора.

Матрица неточностей – это матрица размера $N \times N$, где N — это количество классов. Столбцы этой матрицы резервируются за экспертными решениями, а строки за решениями классификатора. Когда мы классифицируем документ из тестовой выборки мы инкрементируем число стоящее на пересечении строки класса который вернул классификатор и столбца класса к которому действительно относится документ.

1.8.4 F-мера

Понятно что чем выше точность и полнота, тем лучше. Но в реальной жизни максимальная точность и полнота не достижимы одновременно и приходится искать некий баланс. Поэтому, хотелось бы иметь некую метрику которая объединяла бы в себе информацию о точности и полноте нашего алгоритма. В

этом случае нам будет проще принимать решение о том какую реализацию запускать в production (у кого больше тот и круче). Именно такой метрикой является F-мера[1].

F-мера представляет собой гармоническое среднее между точностью и полнотой. Она стремится к нулю, если точность или полнота стремится к нулю.

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

Данная формула придает одинаковый вес точности и полноте, поэтому F-мера будет падать одинаково при уменьшении и точности и полноты. Возможно рассчитать F-меру придав различный вес точности и полноте, если вы осознанно отдаете приоритет одной из этих метрик при разработке алгоритма.

$$F = (\beta^2 + 1)\frac{Precision \times Recall}{\beta^2 Precision + Recall}$$

где β принимает значения в диапазоне $0 < \beta < 1$ если вы хотите отдать приоритет точности, а при $\beta > 1$ приоритет отдается полноте. При $\beta = 1$ формула сводится к предыдущей и вы получаете сбалансированную -меру (также ее называют F_1).

F-мера является хорошим кандидатом на формальную метрику оценки качества классификатора. Она сводит к одному числу две других основополагающих метрики: точность и полноту.

1.8 Эффект переобучения

К сожалению, в имеющемся в нашем распоряжении наборе данных (описанном в разделе 4.1), лишь небольшое число объектов. В таком случае велик риск ситуации, называемой переобучением. Поэтому, рассмотрим эффект переобучения более подробно. Приведенное ниже описание выполнено с помощью статьи [15].

1.8.1 Понятие переобучения

Переобучение (*переподгонка*, *пере-* в значении «слишком», англ. *overfitting*) в машинном обучении и статистике — явление, когда построенная модель хорошо объясняет примеры из обучающей выборки, но относительно плохо работает на примерах, не участвовавших в обучении (на примерах из тестовой выборки).

Это связано с тем, что при построении модели («в процессе обучения») в обучающей выборке обнаруживаются некоторые случайные закономерности, которые отсутствуют в генеральной совокупности.

Даже тогда, когда обученная модель не имеет чрезмерного количества параметров, можно ожидать, что эффективность её на новых данных будет ниже, чем на данных, использовавшихся для обучения. В частности, значение коэффициента детерминации будет сокращаться по сравнению с исходными данными обучения.

1.8.2 Способы борьбы с переобучением

Способы борьбы с переобучением зависят от метода моделирования и способа построения модели. Например, если строится дерево принятия решений, то можно обрезать некоторые его ветки в процессе построения.

Для того чтобы избежать чрезмерной подгонки, необходимо использовать дополнительные методы, например:

- перекрёстная проверка,
- регуляризация ,
- ранняя остановка,
- вербализация нейронных сетей,
- априорная вероятность,
- байесовское сравнение моделей (англ. *bayesian model comparison*),

которые могут указать, когда дальнейшее обучение больше не ведёт к улучшению оценок параметров. В основе этих методов лежит явное ограничение на сложность моделей, или проверка способности модели к обобщению путём оценки её эффективности на множестве данных, не использовавшихся для обучения и считающихся приближением к реальным данным, к которым модель будет применяться.

1.9 Оптимизация гиперпараметров

Зачастую, алгоритм машинного обучения имеет набор параметров (который называется гиперпараметрами), от правильности задания которого зависит качество результирующей модели. Поэтому, для того, чтобы оценить, какой метод машинного обучения лучше, гиперпараметры подбираются достаточно близко к оптимальным (в той мере, в которой это удалось), и далее сравнивают алгоритмы, инициализированные найденными наилучшими параметрами. Поэтому, для того,

чтобы сравнить алгоритмы верно, необходимо уметь правильно выбирать гиперпараметры. Поэтому, рассмотрим основные методы, позволяющие выбрать гиперпараметры близко к оптимальным значениям. На самом деле, задача поиска гиперпараметров, близких к оптимальным можно рассматривать как задачу оптимизации. Однако, к сожалению, большинство оптимизационных методов накладывают различные (причем достаточно жесткие) ограничения, которые не выполняются в случае оптимизации гиперпараметров. Поэтому, методы оптимизации гипермараметров рассматривают целевую функцию как черный ящик, то есть, не накладывают на нее серьезных ограничений (как, например, выпуклость, унимодулярность, гладкость). К сожалению, это существенно сказывается на теоретических и практических свойствах таких алгоритмов. Самое важное, что стоит отметить, так это то, что такие алгоритмы не гарантируют, что найденный результат будет оптимальным с некоторой точностью. Однако на практике оказывается, что результат хороших алгоритмов оказывается приемлемым и в большинстве случаев не уступает по качеству ручному поиску параметров, несмотря на то, что ручной поиск обладает тем важным преимуществом, что при подборе параметров исследователь уже имеет опыт решения различных задач, поэтому имеет примерное представление о том, как подобрать параметры правильно.

1.9.1 Поиск по сетке

Наиболее традиционным способом оптимизации гиперпараметров довольно долгое время был и все еще является поиск по сетке. Поиск по сетке представляет собой исчерпывающий поиск среди параметров, заданных на некоторой сетке (вообще говоря, не равномерной). Среди всех рассматриваемых значений гиперпараметров выбирается то значение, на котором метрика качества максимальна. Метрика качества измеряется с помощью подходящего алгоритма тестирования модели (например, с помощью скользящего контроля). Так как, вообще говоря, полный исчерпывающий поиск всех возможных значений параметров выполнить не всегда возможно (например, потому что возможных значений гиперпараметров бесконечное число), сетку, на которой параметры будут перебираться, необходимо задать заранее.

Например, рассмотрим задачу подбора гиперпараметров градиентного бустинга над решающими деревьями. Одними из основных параметров являются скорость обучения ($learning\_rate$) и количество деревьев ($n\_estimators$). Ограничим возможные значения параметров следующей сеткой:

$$learning\_rate \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$$

$$n\_estimators \in \{10, 20, 50, 100, 200, 500, 1000\}$$

Всего в нашей сетке будет 70 различных вариантов гиперпараметров. Далее, поиск по сетке представляет собой перебор всех этих 70 параметров с последующей оценкой каждого из них (например, с помощью скользящего контроля).

1.9.2 Случайный поиск

Случайный поиск, как и поиск по сетке, представляет собой перебор некоторого множества вариантов гиперпараметров, только, в отличие от поиска по сетке, эти варианты генерируются из некоторого распределения, заданного на вход алгоритму. В простейшем случае, распределение может быть равномерным (равновероятным) на сетке. Так как в случае дискретных распределений есть ненулевая вероятность того, что один и тот же вариант гиперпараметров будет сгенерирован дважды. Чтобы этого избежать, повторно встречающиеся варианты пропускаются. В отличие от поиска по сетке, случайный поиск обладает рядом преимуществ:

1. Количество перебираемых вариантов – один из параметров алгоритма. Можно выбрать его в зависимости от имеющихся вычислительных ресурсов.

2. Зачастую оказывается, что некоторые параметры модели влияют на её качество незначительно (или не влияют вовсе). В таком случае, поиск по сетке будет работать в несколько раз дольше по сравнению с аналогичным поиском, при котором незначимый параметр зафиксирован (то есть, не перебирается), при этом, результаты в двух случаях будут близки. Случайный поиск при этом будет работать в двух вышеописанных случаях с одинаковой производительностью, при этом выдаст аналогичный результат.

3. Иногда бывают ситуации, когда близкие к оптимальным параметры нужно получить до того, как алгоритм отработал до конца (например, чтобы “прикинуть”, на какое качество можно рассчитывать, или в случае сбоя системы). В этом случае, даже результат промежуточного шага будет относительно неплохим, в отличие от результата поиска по сетке. Это связано с тем, при случайном выборе варианта гиперпараметров многие параметры к некоторому шагу примут довольно разнообразное число значений, в отличие от ситуации с

поиском по сетке, при котором некоторые параметры могли принимать фиксированное значение во всех перебранных вариантах.

1.9.3 Выбор “равномерного” множества вариантов гиперпараметров

Если быть точным, то это не метод, а целый класс методов. Основной принцип подобных алгоритмов заключается в том, что вбираются некоторые варианты гиперпараметров так, чтобы, аппроксимация метрики качества была как можно лучше. Например, таковым является метод «латинский гиперкуб».

1.9.4 Байесовская оптимизация

Методы, основанные на байесовской оптимизации являются адаптивными, отличие которых от всех рассмотренных ранее методов заключается в том, что выбор вариантов гиперпараметров для проверки осуществляется с использованием результатов предыдущих проверок.

В данном случае мы решаем задачу максимизации метрики качества. При байесовском подходе искомая метрика качества рассматривается как случайная функция, для которой задано некоторое априорное распределение. Далее, при вычислении значения этой функции в некоторой точке, вычисляется апостериорное распределение этой случайной функции, которое в дальнейшем используется в качестве априорного. При выборе следующей точки используют некоторый критерий и априорное распределение на значение рассматриваемой случайной функции (самым распространенной метрикой в этом случае является “матожидание улучшения качества”).

В отличие от случайного поиска, использование результатов предыдущих вычислений позволяет существенно ускорить сходимость. Правда, недостатком метода является вычислительно более сложный выбор следующей точки, что не подходит в случаях, когда метрика качества вычисляется очень быстро (в таком случае лучше воспользоваться случайным поиском).

1.10 Составление сложных признаков

Заметим, что в исходной задаче одиночные признаки довольно примитивны (это мутации в конкретных участках генома). Возможно, есть смысл рассматривать признаки не по отдельности, а в некоторых комбинациях. Неявно мы уже делаем это, используя решающие деревья, но, может, есть способ найти более релевантные комбинации признаков.

1.10.1 Полиномиальные признаки

На данный момент, для составления сложных признаков используют так называемые полиномиальные признаки. Например, к имеющимся признакам добавляются все попарные произведения признаков. В таком случае, количество признаков возрастает очень сильно, поэтому, необходим интеллектуальный подбор признаков, на котором будет производиться обучение.

1.10.2 Модели, учитывающие комбинации признаков

Некоторые модели (например, ядерный SVM, или методы, основанные на деревьях решений) используют комбинации признаков, поэтому, их можно использовать, не выделяя сложные признаки самостоятельно.

1.10.3 Модели на основе глубокого обучения

В некоторых задачах признаковое описание объектов можно получить с помощью глубинного обучения. Для этого обучают нейросетевую модель, при этом обученная модель может решать, исходную задачу (если таковая имеется), либо некоторую вспомогательную задачу (так, например, сделано в известной модели word2vec). При этом отклик нейронной сети не некотором слое (например, в качестве такого слоя часто выступает предпоследний) содержит достаточно информативное (но, к сожалению, зачастую не интерпретируемое) признаковое описание объекта. При этом такое описание получено достаточно сложными математическими преобразованиями исходных признаков. Недостатком методов на основе глубинного обучения является переобучение модели и необходимость контроля переобучения даже при внушительных объемах данных.

Выводы

Различные методы и алгоритмы машинного обучения позволяют решать широкий круг задач. В том числе, поиск наиболее влияющих на устойчивость к лекарственным препаратам мутаций, построение моделей, использующих эти мутации. Однако, для построения хорошей модели необходимы исследование, сравнение множества различных подходов, а также умелое их комбинирование.

Глава 2. Основанный на переборе метод построения сложных признаков

2.1 Описание алгоритма

Далее, опишем алгоритм, позволяющий искать составные комбинации признаков.

Будем считать, что объект x обладает бинарным признаком a, если значение признака a в объекте x истинно. Также будем считать, что объект x обладает составным признаком $(a_1, a_2, \dots, a_k)$, где $a_1, \dots, a_k$ – исходные бинарные признаки, если он обладает всеми признаками $a_1, \dots, a_k$ (в исходных терминах, это означает, что объект обладает всеми перечисленными мутациями).

На первый взгляд кажется, что все перебрать все возможные составные признаки не представляется возможным. Обоснование этого факта заключается в том, что общее количество признаков составляет несколько тысяч, а перебор всех подмножеств такого числа признаков является трансвычислительной задачей (общее число обработанных бит для решения такой задачи больше, чем число бит, обрабатываемых гипотетическим компьютером размером с Землю за период времени, равный общему времени существования Земли). Однако рассматриваемая задача имеет некоторые особенности, позволяющие перебор всех возможных комбинаций:

1. Признаки сильно коррелированы (в том числе, есть признаки, встречающиеся у одного и того же подмножества объектов).

2. В выборке мало объектов (до 136).

3. При определении составного признака важно не то, какие простые признаки он включает, а какие объекты обладают составным признаком.

4. Большинством исходных признаков обладает малое число объектов (матрица объекты-признаки сильно разрежена).

Используя вышеперечисленные особенности, опишем далее алгоритм, позволяющий перебрать (в некотором смысле) все составные признаки.

Во-первых, используя свойство 4, оговоримся считать составные признаки одинаковыми, если ими обладает одно и то же множество объектов. Используя это, далее будем задавать составной признак не множеством исходных признаков, из

которых он состоит, а множеством объектов, которыми он обладает. В связи со свойством 4, это обеспечит нам гораздо меньший размер результирующего множества составных признаков (ведь в таком случае один и тот же составной признак могут порождать различные исходные признаки). То есть, такое определение идентичности составных признаков позволяет нам строить алгоритм, оперирующий гораздо меньшим набором составных признаков, что позволяет значительно сэкономить как время, так и память (в нашем случае, расходование памяти сокращается до размеров, с которым способна работать имеющаяся в наличии вычислительная система).

Далее, будем определять составной признак множеством объектов $\{x_1, \dots, x_k\}$, обладающих им (такое множество объектов будем называть определяющим). В таком случае введенное нами равенство составных признаков эквивалентно равенству определяющих множеств. Такой способ определения составного признака прост в вычислениях, но, к сожалению, менее интерпретируем, также открытым остается вопрос, как, имея составной признак, заданный с помощью определяющего множества, описать множеством исходных признаков.

Понятно, что для каждого исходного признака легко вычислить его определяющее множество. Посмотрим, как вычислить определяющее множество составного признака, заданного исходными признаками $a_1, \dots, a_k$, имея их определяющие множества $A_1, \dots, A_k$. Вспомнив определение составного множества, нетрудно понять, что определяющим множеством составного признака $(a_1, \dots, a_k)$ является множество $A = A_1 \cap A_2 \cap \dots \cap A_k$.

Далее, заметим некоторые соотношения, позволяющие построить алгоритм, перебирающий все возможные определяющие множества. Пусть X_i – множество всех определяющих множеств составных признаков, в состав которых входят исключительно исходные признаки $a_1, \dots, a_i$. Тогда:

1. $X_0 = \emptyset$
2. $X_{i+1} = X_i \cup \{A_{i+1}\} \cup IntersectEvery(X_i, A_{i+1}), i \in [0, m-1]$

(8)

где m – число признаков, а $IntersectEvery(A, b)$ – множество пересечений элементов A с b.

Заметим, что искомое множество определяющих множеств есть X_m. Далее, из вышеописанных соотношений, следует естественный алгоритм поиска всех определяющих множеств:

1. Инициализируем $X_0 = \emptyset$
2. Пересчитываем X_{i+1} через X_i u A_{i+1} по формуле (8).

К сожалению, приведенный выше алгоритм не обладает приятными теоретическими свойствами (нетрудно придумать такое семейство входных данных, что мощность итогового множества определяющих множеств экспоненциально зависит от числа признаков, а поэтому и алгоритм не может быть полиномиальным). Однако, его временную сложность можно записать в виде $O(AnsSize * m * n)$, где $AnsSize = |X_m|$,m-число признаков, n – число объектов, а сложность по памяти можно записать как $O(AnsSize * n)$ поэтому алгоритм является достаточно эффективным в задачах, размер ответа в которых невелик. В нашем случае это значение порядка 10^8, что позволяет вычислить X_m в течение часа на вычислительной системе с 4гб RAM и процессором Intel Core 2 Duo T6600 с тактовой частотой 2.2Ггц. Также заметим, что при применении вышеописанного алгоритма мы пока никак не использовали скрытые переменные. Это позволяет провести вычисление X_m лишь один раз, и далее использовать ответ при различных скрытых переменных (что в нашем случае позволяет использовать один раз посчитанные результаты для работы с различными лекарствами).

2.2 Оценка полученных составных признаков

Далее, опишем некоторые подходы, которые можно использовать для ранжирования релевантности составных признаков. В отличие от раздела 2.1, на данном этапе у нас определены скрытые переменные y_i и нам необходимо отобрать из них те, которые мы далее добавим к текущим признакам. Мы предлагаем несколько стратегий.

2.2.1 “Простая” стратегия

“Простая” стратегия заключается в том, что мы сначала оставляем только те составные признаки, скрытые переменные всех элементов определяющего множество которых имеет значение «истина» (то есть, объект устойчив к препарату). Далее, полученные признаки ранжируем по мощности определяющего множества (чем больше мощность, тем выше признак в итоговом списке) и выбираем признаки из начала списка (количество признаков является параметром алгоритма).

2.2.2 Стратегия, основанная на оценке значимости признака с использованием бета-биномиальной статистической модели

Бета-биномиальная модель – двухуровневая байесовская модель, позволяющая осуществлять оценку параметра p биномиального распределения $B(n,p)$. В отличие от частотной оценки, оценка с использованием бета-биномиальной модели разумно даже в случае небольшой выборки. В случае частотной оценки параметр p считается независимым параметром, что приводит к неточным оценкам в случае небольших выборок. В бета-биномиальной модели параметр p считается случайной величиной и имеет априорное распределение $Be(a,b)$. Параметры a,b могут быть как параметрами алгоритма, так и подбираться по данным с использованием метода максимальной обоснованности. Для этого необходимо решить задачу выпуклой оптимизации (например, методом Нелдера-Мида).

С помощью бета-биномиальной модели можно оценить вероятность того, что если значение признака положительно, объект устойчив к лекарству, используя распределение известных скрытых переменных у объектов, обладающих признаком. Далее, достаточно выбрать признаки с наибольшим значением оцененной вероятности.

Более подробно о бета-биномиальной модели можно почитать в статье [4].

2.3 Получение составных признаков для новых объектов

Ранее, мы определяли составной признак через определяющие множества. Это было достаточно удобно для вычислений, однако представление составного признака в такой форме не даёт непосредственной возможности получить значение признака для нового объекта, ведь его нет ни в одном из определяющих множеств. Здесь мы предлагаем несколько вариантов решения этой проблемы.

1. Новый объект обладает составным признаком, если он обладает всеми общими признаками объектов определяющего множества.

2. Для составного признака найдем минимальное по размеру множество простых (исходных) признаков, таких, что определяющее множество составного признака, полученное из признаков искомого множества, совпадает с данным определяющим множеством.

3. Будем считать, что тестовый объект нам дан еще до стадии обучения (при этом, чтобы тестирование было честным, нам будет известно только

признаковое описание). Тогда, чтобы получить значение составного признака, достаточно проверить принадлежность данного объекта определяющему множеству рассматриваемого составного признака.

2.4 Программная реализация

Перебор составных признаков был реализован на языке C++ с использованием библиотеки для эффективной работы с хеш-таблицами google sparse-hash [16]. Такой выбор обоснован тем, что приходится оперировать большим числом определяющих множеств (порядка 10^8), что накладывает сильные ограничения на эффективность использования памяти, что является одними из основных преимуществ выбранных технологий. При помощи библиотек boost-python и boost-numpy реализовано расширение написанного модуля перебора для языка python. Это было необходимо для совместимости с остальными модулями (написанными на Python). Различные стратегии по использованию составленных признаков реализованы на C++ и Python. Выбор зависел от их производительности. Метод Нелдера-Мида взят из библиотеки scipy.optimize.

Выводы

В некоторых случаях особенности задачи могут создать хорошие предпосылки для разрешимости относительно них некоторых вспомогательных задач. В нашем случае, вспомогательной задачей является перебор составных признаков. Однако, к сожалению, некоторые вопросы остаются все еще не решенными. Например, как правильно выбрать из всех составных признаков самые важные (вероятно, предложенные способы являются далеко не самыми оптимальными).

Глава 3. Окружение для тестирование различных моделей

Данная нам задача анализа лекарственной устойчивости обладает рядом своих особенностей. Среди них малое число объектов и большое число признаков. К нашему сожалению, именно в таких ситуациях велик риск переобучения. В связи с этим, чтобы полученным результатам можно было доверять, важно построение правильного окружения, позволяющее проводить различные эксперименты.

3.1 Требования к окружению тестирования моделей

Опишем основные требования к тестирующему окружению, которых будем придерживаться при его проектировании и реализации:

1. Независимость от гиперпараметров. То есть, для наших экспериментов мы задаём лишь структура нашей модели, но не указываем конкретные значения параметров. Это важно для того, чтобы тестировать структуру моделей а не умение подбора параметров.

2. Подбор параметров эксперимента проводится независимо от тестирования. Это означает, что тестовые данные не участвуют при подборе гиперпараметров. То есть, тестирование должно проводиться ровно один раз, после того, как параметры перебраны. Это важно, потому что, если, перебирая параметры моделей (ручным или автоматическим способом), мы будем иметь информацию о качестве модели на тестовых данных, очень просто переобучиться. Обычно, этому уделяют не так много внимания, потому что в выборках достаточно много объектов и значительные изменения качества модели с большой вероятностью свидетельствуют именно о лучшем качестве, а не о переобучении.

3. Устойчивость. Это означает, что оценка качества должна слабо зависеть от разбиения выборки на обучающую и тестовую.

3.2 Архитектура тестирующей системы

В связи с этими требованиями, построение тестирующей системы мы решили производить следующим образом.

1) На вход поступает данные по некоторому лекарству, модель, а также пространство неинициализированных параметров этой модели.

2) Используя эту модель и пространство параметров мы создаем так называемую мета-модель, у которой уже нет никаких параметров, кроме переданной модели и пространства возможных значений параметров (на самом

деле, мета-модель как раз инкапсулирует в себе подбор этих самых гиперпараметров).

3) Мета-модель тестируется на имеющихся данных. На выход поступают результаты тестирования – метрики качества мета-модели и отобранные ею признаки.

Визуальное описание архитектуры тестирующего окружения можно видеть на рисунке 1.

Рисунок 1 - структура тестирующего окружения

Тестирование мета-модели проводилось посредством скользящего контроля по 5 блокам. В качестве метрик качества использовались правильность, F_1, точность, полнота, матрица неточностей.

Заметим, что такой способ тестирования обеспечивает нам выполнения сразу трех пунктов. Во-первых, так как тестируется мета-модель, выполняется требование 1, ведь мета-модель не зависит от гиперпараметров. Так как мета-модель сама является моделью, а тестирование методом скользящего контроля по 5 блокам удовлетворяет условиям 2 и 3, они также выполнены.

3.3 Устройство мета-модели

Мета-модель представляет собой модель, которая принимает на вход другую модель и пространство её гиперпараметров, подбирает наилучшие гиперпараметры и использует их для обучения и предсказания. Рассмотрим это более детально.

3.3.1 Обучение мета-модели

На вход поступают тренировочные данные – признаковое описание объектов и скрытые переменные. Обучение модели состоит из следующих последовательных стадий:

4) Подбор оптимальных гиперпараметров с помощью скользящего контроля по 5 блокам. Для подбора гиперпараметров использовалась библиотека hyperopt, реализующая метод дерева парзеновских оценок (Tree of Parzen Estimators), метод байесовской оптимизации гиперпараметров.

5) Обучение внутренней модели с использованием гиперпараметров, подобранных на шаге 1.

3.3.1 Предсказание мета-модели

На вход поступает тестовые данные, представленные признаковым описанием тестовых объектов. Скрытые переменные при этом неизвестны. В этом случае производится предсказание скрытых переменных с использованием обученной на стадии обучения внутренней модели.

3.4 Оценка значимости результатов

Для оценки значимости различий между качеством моделей использовался следующий статистический тест. Пусть y_{true} – скрытые бинарные метки классов, y_1-предсказания первой модели, а y_2-предсказания второй модели, полученные посредством скользящего контроля. Далее, построим следующую величину: $z_i = [y_{true_i} = y_{1_i}] - [y_{true_i} = y_{2_i}]$, после чего, на основании величин $z_1, \dots, z_k$, проверялась статистическая гипотеза $Ez = 0$ с помощью приближения среднего арифметического мультиномиальных величин нормальной. Если гипотеза отклоняется, различие между качеством моделей считается статистически значимым.

3.5 Программная реализация

Весь исходный код написан на языке Python. Оптимизация гиперпараметров выполнена с помощью библиотеки hyperopt [17]. Реализация скользящего

контроля и различных метрик качества алгоритмов выполнены с помощью scikit-learn [18]. Проверка статистической гипотезы (при проверке значимости результатов) реализована при помощи библиотеки scipy [19].

Выводы

Особенности задачи играют важную роль не только на способ и сложность решения самой задачи, но и на способ тестирования результатов, который приходится сильно усложнять по сравнению с способами, отлично применяемыми в большом числе других задач.

Глава 4. Исходные данные и проведенные эксперименты

4.1 Исходные данные

Далее, опишем исходные данные, имеющиеся у нас для проверки различных алгоритмов отбора признаков.

По итогам анализа имеющихся данных полногеномного секвенирования M.tuberculosis и медицинских карт пациентов были отобраны 132 генома и сформированы тестовые наборы данных (табл. 1). Чувствительность проверялась к препаратам так первой и второй линии. Так как препараты первой линии и устойчивость к ним хорошо изучена, далее мы будем использовать исключительно препараты второй линии. Также стоит отметить, что в данных присутствуют пропуски – не удалось достоверно проверить наличие точечной мутации.

Линия	Название препарата	Число наблюдений
1	Ethambutol	136
	Isoniazid	136
	Rifampicin	134
	Rifampentine	2
	Pyrazinamide	36
	Streptomycin	136
	Cycloserine	129
2	Ethionamide/Prothionamide	136
	Para-aminosalicyclic acid	136

	Capremycin	131
	Amikacin	136
	Ofloxacin	136
	Kanamycin	17

Таблица 2 - краткое описание исходного набора данных

Препараты первой линии хорошо изучены, поэтому особый интерес представляют препараты второй группы. Мы будем рассматривать все препараты второй группы, кроме “Kanamycin”, так как представлено всего 17 объектов, причем только 2 не являются лекарственно устойчивыми.

4.2 Основные примитивы, использованные при составлении экспериментов

Для того чтобы при задании сложных экспериментов не приходилось задавать все возможные варианты параметров, мы решили объединить относительно несложные пары (модель, пространство параметров) в примитивы, из которых в дальнейшем строим свои эксперименты. Это оказалось очень удобно. Например, если мы ходим в нашем эксперименте использовать один из нескольких базовых алгоритмов машинного обучения (но пока сами не знаем, какой из них работает лучше всего), можно воспользоваться примитивом “базовый алгоритм”, вместо указания, из каких именно моделей мы хотим выбирать, какие у них могут быть параметры и так далее. Примитивы могут быть инициализированы как полностью, так и частично (то есть, могут содержать параметры, требующие инициализации). Это мы используем в случае, когда модель представляет собой декоратор другой модели, декорируемая модель выступает в качестве параметра.

4.2.1 Примитив “выбор”

Примитив “выбор” мы используем в случае, когда у нас есть несколько вариантов модели, но мы не знаем наверняка, какой из них лучше. В таком случае мы используем примитив “выбор”, параметрами которого являются различные варианты.

4.2.2 Примитив “логистическая регрессия”

В качестве модели примитива “логистическая регрессия” выступает логистическая регрессия. Перебираемые параметры при этом следующие:

1. Коэффициент регуляризации (от e^{-15} до e^{15} с использованием логарифмической шкалы).

2. Метод регуляризации (l_1 или l_2).

3. Веса классов (единичные или сбалансированные).

4.2.3 Примитив “случайный лес”

В качестве модели выступает случайный лес со 100 деревьями. Случайный лес зарекомендовал себя как очень устойчивый классификатор, поэтому мы не перебираем его параметры.

4.2.4 Примитив “градиентный бустинг”

В качестве модели используется классификатор XGBClassifier библиотеки xgboost. Перебираемые параметры при этом следующие:

1. Количество деревьев (от 1 до 200).

2. Максимальная глубина дерева (от 1 до 13).

3. Минимальное число листовых объектов (от 1 до 6).

4. Размер поднабора данных, используемый при обучении одного дерева (от половины всех данных до полного набора).

5. Минимальное изменение функции потерь, необходимое для разделения листового узла (от 0.5 до 1).

Более подробное их описание можно почитать в документации xgboost [7].

4.2.5 Примитив “простая модель”

В качестве примитива “простая модель” выступает выбор из логистической регрессии, случайного леса и градиентного бустинга.

4.2.6 Примитив “статистический метод отбора признаков”

В качестве примитива выступает выбор из двух методов, основанных на проверке статистических гипотез: хи-квадрат и ANOVA. Также примитив декорирует модель, использующую далее отобранные признаки. Перебираем следующие параметры:

1. Статистический критерий (хи-квадрат или ANOVA)

2. Количество результирующих признаков (от e^0 до e^5 по логарифмической шкале).

4.2.7 Примитив “модель для оценки признаков ”

В качестве примитива мы использовали выбор из случайного леса и логистической регрессии.

4.2.8 Примитив “устойчивая простая модель”

Абсолютно аналогично примитиву “модель для оценки признаков”.

4.2.9 Примитив “отбор признаков на основе модели”

В качестве примитива выступает метод, отбирающий признаки на основе важности, которую вычисляет примитив “модель для оценки признаков”. В качестве параметров выступает модель, вычисляющая порог, и декорируемая модель, использующая отобранные признаки.

4.2.10 Примитив “отбор признаков”

В качестве примитива используется выбор из отбора признаков на основе модели и статистического метода отбора признаков.

4.2.11 Примитив “сеть релевантных признаков”

В качестве примитива используется, как нетрудно догадаться, сеть релевантных признаков. При этом, перебираются следующие параметры:

1. λ (от e^{-5} до e^{15} по логарифмической шкале). Подробное описание параметра можно почитать в разделе 1.6.

2. α (от 0 до 1). Подробное описание параметра можно почитать в разделе 1.6

4.2.12 Примитив “составление сложных признаков”

В качестве примитива используется модель составления сложных признаков, описанная в разделе 1.10. Перебираются следующие параметры:

1. Способ задания приоритетов: “простой” способ и способ, основанный на бета-биномиальной байесовской модели.

2. Минимальное число объектов определяющего множества, необходимое для того, чтобы использовать признак (от 0 до 10).

3. Способ присвоения признака новому объекту (с использованием тестового примера на этапе обучения, с использованием всех общих признаков,

или с использованием минимального числа признаков, порождающих определяющее множество).

4. Количество добавляемых признаков (от 0 до e^{10} с использованием логарифмической шкалы).

Далее, мы будем использовать лишь высокоуровневые примитивы “обогащение признаками”, “сеть релевантных признаков”, “базовая модель”, “устойчивая простая модель”, “простая модель”, “отбор признаков”.

4.3 Предварительная обработка данных

Перед тем, как передавать данные непосредственно в модели, мы выполнили предварительную обработку данных, которая включает:

1. Пропуски заменяются нулями (что эквивалентно отсутствию мутаций). Это кажется разумным решением, так как 0 – самое частое значение в матрице, причем сильно преобладает, при этом пропуск наверняка неинформативен.

2. Удаляются признаки, которые встретились в данных меньше трех раз, а также удаляются признаки, которые встретились больше, чем в половине случаев (ведь, если мутация встречается часто, это означает, что это скорее распространенное явление, чем патология).

4.4 Описание выполненных экспериментов

4.4.1 Простая модель

Эксперимент представляет собой запуск примитива “простая модель” на предварительно обработанных данных.

4.4.2 Отбор признаков и модель

Эксперимент представляет собой последовательный запуск примитивов “отбор признаков” и “простая модель” на предварительно обработанных данных.

4.4.3 Сеть релевантных признаков и модель

Эксперимент представляет собой последовательный запуск примитивов “сеть релевантных признаков” и “простая модель” на предварительно обработанных данных.

4.4.4 Составление сложных признаков и устойчивая простая модель

Эксперимент представляет собой последовательный запуск примитивов “составление сложных признаков” и “устойчивая простая модель” на предварительно обработанных данных.

4.4.5 Составление сложных признаков, отбор признаков и простая модель

Эксперимент представляет собой последовательный запуск примитивов “составление сложных признаков”, “отбор признаков” и “ простая модель” на предварительно обработанных данных.

4.4.6 Составление сложных признаков, сеть релевантных признаков и простая модель

Эксперимент представляет собой последовательный запуск примитивов “составление сложных признаков”, “сеть релевантных признаков” и “ простая модель” на предварительно обработанных данных.

4.5 Программная реализация

Весь исходный код предварительной обработки данных, проектирования примитивов и экспериментов реализованы на языке Python. Чтение и предварительная обработка данных выполнена с помощью библиотек pandas [20] и numpy [21]. Логистическая регрессия и случайный лес взяты из библиотеки scikit-learn [18], градиентный бустинг взят из библиотеки xgboost [7]. Примитивы реализованы при помощи библиотеки hyperopt [17].

4.6 Результаты и их анализ

На таблице 2 отображены результаты проведенных экспериментов на различных наборах данных. Жирным шрифтом выделены наилучшие значения среди всех проведенных экспериментов с препаратом. Так как признаки не интерпретируемы (это просто номера мутаций), поместим их описание в приложение 1.

		AMIK: Amikacin	**CAPR: Capreomycin**	**ETHI: Ethionamide/ Prothionamide**	**OFLO: Ofloxacin**	**PARA: Para-aminosalicyclic acid**
Accuracy	**4.4.1**	0,89	0,83	0,75	0,83	**0,88**
	4.4.2	0,89	0,82	**0,79**	**0,87**	**0,88**
	4.4.3	0,88	0,82	0,76	0,85	**0,88**
	4.4.4	0,89	0,82	0,76	0,82	**0,88**
	4.4.5	0,81	**0,85**	0,71	0,82	**0,88**
	4.4.6	**0,91**	0,83	0,74	0,82	0,87

F1	**4.4.1**	0,88	0,83	0,37	0,84	**0,6**
	4.4.2	0,88	0,83	**0,54**	**0,88**	0,59
	4.4.3	0,87	0,82	0,38	0,85	0,58
	4.4.4	0,88	0,83	0,33	0,84	0,59
	4.4.5	0,77	**0,84**	0,41	0,84	0,59
	4.4.6	**0,9**	0,83	0,28	0,83	0,55
TN	**4.4.1**	67	54	92	54	108
	4.4.2	68	53	90	**55**	107
	4.4.3	67	51	94	54	**109**
	4.4.4	67	50	**96**	51	107
	4.4.5	67	**58**	82	51	107
	4.4.6	**70**	56	93	50	107
FP	**4.4.1**	4	5	8	6	4
	4.4.2	**3**	6	10	**5**	5
	4.4.3	4	8	6	6	**3**
	4.4.4	4	9	**4**	9	5
	4.4.5	4	**1**	18	9	5
	4.4.6	1	3	7	10	5
FN	**4.4.1**	**11**	17	26	17	**12**
	4.4.2	12	17	**19**	**13**	**12**
	4.4.3	12	16	26	15	13
	4.4.4	**11**	**14**	28	15	12
	4.4.5	22	19	22	15	**12**
	4.4.6	**11**	19	29	15	13
TP	**4.4.1**	**54**	55	10	59	**12**
	4.4.2	53	55	**17**	**63**	**12**
	4.4.3	53	56	10	61	11
	4.4.4	**54**	**58**	8	61	**12**
	4.4.5	43	53	14	61	**12**
	4.4.6	**54**	53	7	61	11

Таблица 3 - результаты экспериментов

Как видно из таблицы 2, результаты в целом достаточно похожи. Иногда, некоторые модели сильно деградируют в производительности, например, 4.4.5 на препарате “ETHI: Ethionamide/ Prothionamide”. Среди всех рассмотренных моделей можно выделить модель 4.4.2 (отбор признаков и простая модель), потому что при проверке ее на всех пяти препаратах она выдавала либо наилучшее, либо очень близкое к наилучшему (отличающееся не более чем на 3 пункта) значение правильности и $F1$.

Для демонстрации процесса сходимости, изобразим графики, отражающие правильность алгоритма в процесса сходимости на внутреннем скользящем контроле (в мета-модели) и на внешнем скользящем контроле (при тестировании мета-модели). Так как всего экспериментов проведено достаточно много (а следовательно, и графиков можно построить достаточно много), отобразим

графики только для препарата “ CAPR: Capreomycin ”. На рисунках 2-7 можно видеть результаты экспериментов 4.4.1 – 4.4.6 соответственно.

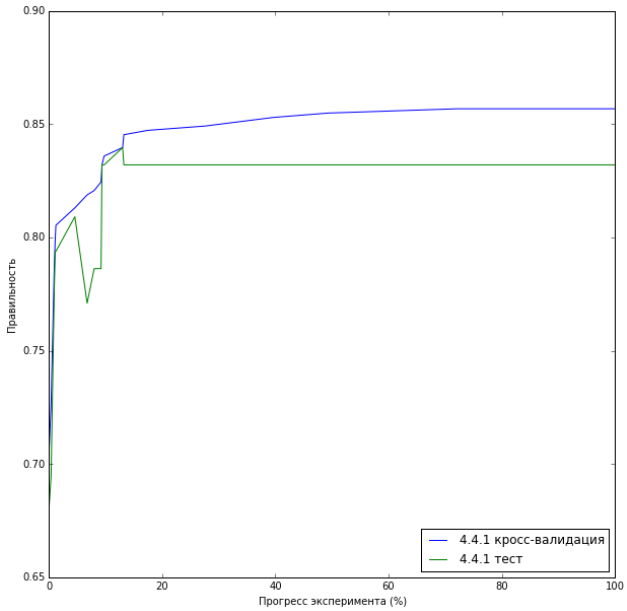

Рисунок 2 - результаты модели 4.4.1

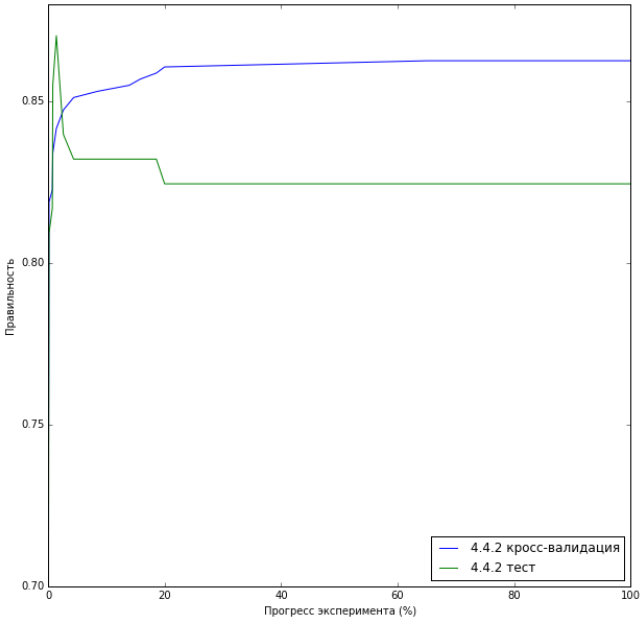

Рисунок 3 - результаты модели 4.4.2

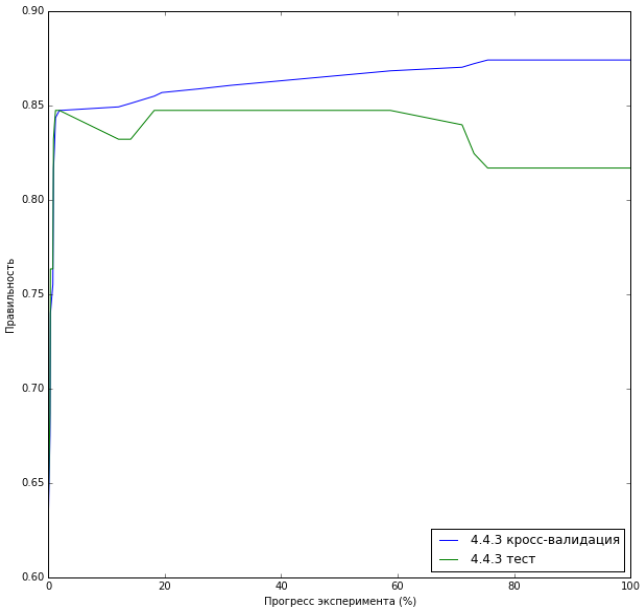

Рисунок 4 - результаты модели 4.4.3

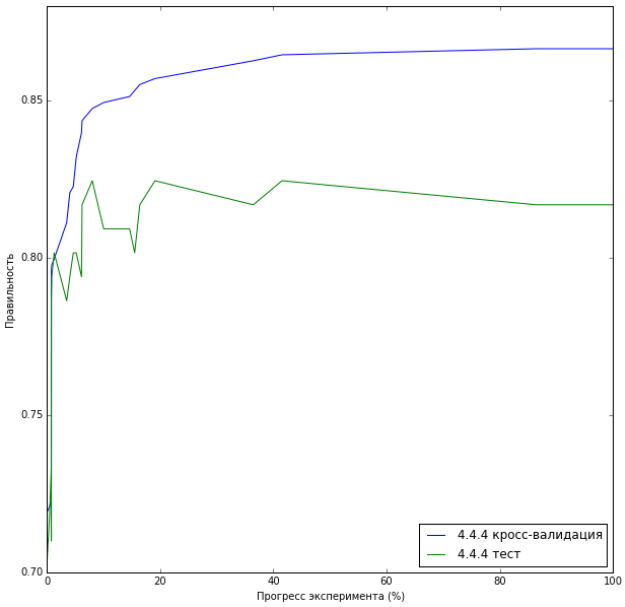

Рисунок 5 - результаты подели 4.4.4

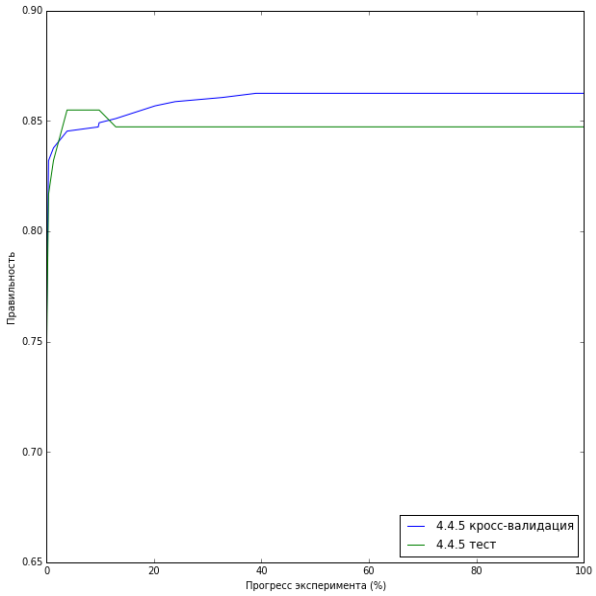

Рисунок 6 - результаты модели 4.4.5

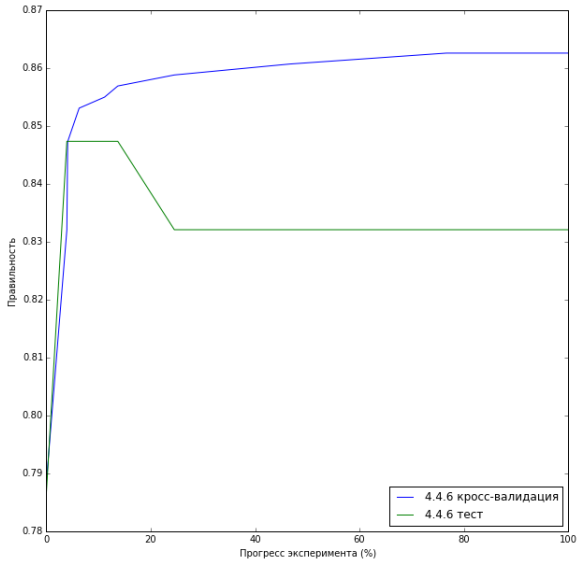

Рисунок 7 - результаты модели 4.4.6

Выводы

Сеть релевантных признаков является достаточно интересным применением аппарата графических моделей к задаче отбора признаков. Однако, к сожалению, в рассмотренной задаче данная модель не показала улучшений по сравнению с базовыми методами отбора признаков. По результатам проведенных экспериментов, можно сказать, что применение базовых методов отбора признаков более предпочтительно. Также, по рисункам 2-7 видно, что при

оптимальном подборе параметров на тренировочных данных посредством скользящего контроля, качество при подборе параметров заметно выше, чем итоговое качество при тестировании. Это говорит об обоснованности построения двухуровневой схемы тестирования для получения объективных результатов в рассматриваемой задаче.

Заключение

В ходе проведенной работы было сделано следующее:

1. Проведен обзор некоторых базовых алгоритмов классификации и отбора признаков.
2. Предложен и реализован метод поиска составных признаков.
3. Рассмотрена и реализована сеть релевантных признаков – метод отбора признаков, основанный на аппарате графических моделей.
4. Реализовано окружение, позволяющее объективно и независимо от параметров моделей тестировать различные методы машинного обучения.
5. С помощью реализованного окружения были проведены эксперименты на реальных наборах данных для исследования связи мутаций (однонуклеотидных полиморфизмов) генома микробактерии и ее лекарственной устойчивости к различным лекарственным препаратам.
6. Произведен анализ результатов экспериментов.

Список использованной литературы

1. Liu, J. High-Dimensional Structured Feature Screening Using Binary Markov Random Fields / J. Liu [et al.]. // JMLR workshop and conference proceedings. – 2012. - №22. – P. 712–721.

2. Bishop, C.M. Graphical Models / C.M. Bishop // Pattern Recognition and Machine Learning. – Springer, 206. - Ch. 8. – P. 359-418.

3. Субмодулярная релаксация в задаче минимизации энергии марковского случайного поля [Электронный ресурс] / Осокин А. – Режим доступа: http://www.machinelearning.ru/wiki/images/8/83/OsokinThesis.pdf - Дата доступа: 23.05.2016

4. Navaro, D. An introduction to the Beta-Binomial model [Электронный ресурс] / D. Navaro – Режим доступа: https://www.cs.cmu.edu/~10701/lecture/technote2_betabinomial.pdf - Дата доступа: 23.05.2016

5. Breiman, Leo (2001). «Random Forests». *Machine Learning* **45** (1): 5–32. DOI:10.1023/A:1010933404324

6. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. // Hastie, T., Tibshirani R., Friedman J. — 2nd ed. — Springer-Verlag, 2009. — 746 p. — ISBN 978-0-387-84857-0.

7. Get Started with XGBoost [Электронный ресурс] / Read the docs documentation. - Режим доступа: https://xgboost.readthedocs.io - Дата доступа: 23.05.2016

8. Логистическая регрессия [Электронный ресурс] / Википедия. – Режим доступа: https://ru.wikipedia.org/wiki/Логистическая_регрессия - Дата доступа: 23.05.2016

9. Дерево принятия решений [Электронный ресурс] / Википедия. – Режим доступа: https://ru.wikipedia.org/wiki/Дерево_принятия_решений - Дата доступа: 23.05.2016

10. Random forest [Электронный ресурс] / Википедия. – Режим доступа: https://ru.wikipedia.org/wiki/Random_forest - Дата доступа: 23.05.2016

11. Бустинг [Электронный ресурс] / Профессиональный информационно-аналитический ресурс, посвященный машинному обучению, распознаванию образов и интеллектуальному анализу данных – Режим доступа: http://www.machinelearning.ru/wiki/index.php?title=Бустинг – Дата доступа: 23.05.2016

12. Графическая вероятностная модель [Электронный ресурс] / Википедия – Режим доступа: https://ru.wikipedia.org/wiki/Графическая_вероятностная_модель - Дата доступа: 23.05.2016

13. Кросс-валидация [Электронный ресурс] / Профессиональный информационно-аналитический ресурс, посвященный машинному обучению, распознаванию образов и интеллектуальному анализу данных – Режим доступа: http://www.machinelearning.ru/wiki/index.php?title=Кросс-валидация – Дата доступа: 23.05.2016

14. Оценка классификатора (точность, полнота, F-мера) [Электронный ресурс] / Баженов Д. - Режим доступа: http://bazhenov.me/blog/2012/07/21/classification-performance-evaluation.html - Дата доступа: 23.05.2016

15. Переобучение [Электронный ресурс] / Википедия – Режим доступа: https://ru.wikipedia.org/wiki/Переобучение - Дата доступа: 23.05.2016

16. Sparsehash [Электронный ресурс] / Github – Режим доступа: https://github.com/sparsehash/sparsehash - Дата доступа: 23.05.2016

17. Hyperopt: Distributed Asynchronous Hyperparameter Optimization in Python [Элекронный ресурс] / Github – Режим доступа: https://github.com/hyperopt/hyperopt - Дата доступа: 23.05.2016

18. scikit-learn: machine learning in Python [Электронный ресурс] / scikit-learn 0.17.1 documentation – Режим доступа: http://scikit-learn.org/ - Дата доступа: 23.05.2016

19. SciPy.org [Электронный ресурс] / SciPy.org – Режим доступа: https://www.scipy.org/ - Дата доступа: 23.05.2016

20. Pandas: Python Data Analysis Library [Электронный ресурс] / pandas – Режим доступа: http://pandas.pydata.org/ - Дата доступа: 23.05.2016

21. NumPy [Электронный ресурс] / Numpy – Режим доступа: http://www.numpy.org/ - Дата доступа: 23.05.2016

Приложение 1