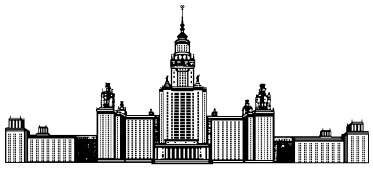
Московский Государственный Университет имени М.В. Ломоносова
Факультет вычислительной математики и кибернетики
Кафедра математических методов прогнозирования



Дипломная работа

Машинное обучение с категориальными признаками

Machine learning with categorical features

Работу выполнил
студент 517 группы
Фонарев Александр Юрьевич

Научный руководитель
д.ф.-м.н., профессор
Дьяконов Александр Геннадьевич

Москва
2014

Содержание

1 Введение

Все более и более востребованным на практике становятся методы нахождения закономерностей в больших объемах данных и, в частности, алгоритмы машинного обучения на прецедентах. Подавляющее большинство таких алгоритмов позволяют учитывать лишь вещественные признаки для описания наблюдаемых объектов. Однако в реальной практике часто встречаются задачи с категориальными признаками, принимающими свои значения из конечного неупорядоченного множества. В настоящей работе проанализированы имеющиеся алгоритмы, учитывающие категориальные признаки, а также предложены новые подходы. Работа всех описанных методов была исследована на реальных наборах данных.

2 Постановка задачи

2.1 Основные понятия

2.1.1 Задача бинарной классификации

Напомним, что такое задача бинарной классификации [21]. Пусть задано множество объектов $\mathcal{X}$, множество меток $\mathcal{Y} = \{0, 1\}$, и существует целевая функция $y^* : \mathcal{X} \to \mathcal{Y}$, значения которой $y_i = y^*(x_i)$ известны только на конечном подмножестве объектов $X_1, ..., X_n \in \mathcal{X}$. Пары «объект-ответ» (X_i, y_i) называются прецедентами. Совокупность пар $(X_i, y_i)_{i=1}^n$ называется обучающей выборкой.

Задача обучения по прецедентам заключается в том, чтобы по обучающей выборке научиться восстанавливать зависимость y^*, то есть построить решающую функцию $\mathcal{X} \to \mathcal{Y}$, которая бы приближала целевую функцию $y^*(x)$, причем не только на объектах обучающей выборки, но и на всем множестве $\mathcal{X}$. Кроме того, решающая функция должна допускать эффективную компьютерную реализацию.

Каждый объект X_i задается измерениями некоторых своих характеристик — признаков. Допустим, признаков всего m штук. Тогда каждому объекту $X_i \in \mathcal{X}$ соответствует вектор $(X_i^1, ..., X_i^m)$ — признаковое описание. Таким образом обучающую выборку можно представить в виде матрицы $X \in \mathbb{R}^{n \times m}$.

2.1.2 Оценка качества алгоритмов

Для оценки качества работы обученных алгоритмов используют различные функционалы [21]. В настоящей работе будет использована метрика AUC (площадь под ROC-кривой). Допустим, есть некоторый истинный вектор меток y для n объектов

и вектор $\tilde{y}$ предсказанных «степеней принадлежности» классу 1. Тогда

$$AUC(y, \tilde{y}) = \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} I[y_i < y_j] \cdot I[\tilde{y}_i < \tilde{y}_j]}{(\sum_{i=1}^{n} I[y_i = 0]) \cdot (\sum_{i=1}^{n} I[y_i = 1])} \in [0, 1].$$

Чем ближе значени AUC к единице — тем выше качество алгоритма.

Метрика AUC обладает несколькими интересными свойствами. Например, она устойчива к монотонным преобразованиям предсказанных меток ответов. Поэтому при решении задач необходимо будет лишь учитывать взаимных порядок объектов по предсказанным «степеням принадлежности».

Обычно для более объективной оценки качества, алгоритм обучается на одной части выборки, а предсказание оценивается на другой. Если разбить выборку на k равных непересекающихся частей (фолдов), то можно поочередно обучаться на $k-1$ части и оценивать качество на оставшейся одной, а полученные в результате k оценок качества рассматривать как наблюдения случайной величины оценки качества. Такой подход называется кроссвалидацией.

2.2 Постановка задач с категориальными признаками

Многие классические методы машинного обучения предполагают, что все признаки $X^j \in \mathbb{R}$. Однако в некоторых задачах признаки могут принимать значения из множеств, не совпадающих с множествами вещественных чисел. Так, например, признаки могут принимать значения из конечного неупорядоченного множества. Например, это может быть признак *Город* со значениями из множества {*Москва, Санкт-Петербург, Новосибирск, Казань, ...*}. Такие признаки называются категориальными, факторными или номинальными.

Казалось бы, можно просто свести задачу с категориальными признаками к задаче с вещественными просто пронумеровав значения признаков. Например,

$$\textit{Москва} \ \rightarrow \ 1,$$

$$\textit{Санкт-Петербург} \ \rightarrow \ 2,$$

$$\textit{Новосибирск} \ \rightarrow \ 3,$$

$$\textit{Казань} \ \rightarrow \ 4,$$

$$\ldots \ .$$

Однако такой подход обычно заканчивается неудачей. Действительно, ведь исходное множество значений неупорядочено, а на пронумерованном множестве задан порядок, который, скорее всего, будет учитывать алгоритм. К тому же, не ясно, какую именно нумерацию использовать, ведь всего существует $q!$ нумераций с различным взаимным порядком, где q — количество уникальных значений признака.

2.3 Примеры задач с категориальными признаками

В последние годы задачи с категориальными признаками приобретают все большую и большую популярность.

Большое семейство задач с категориальными признаками представляют из себя задачи, связанные с коллаборативной фильтрацией. В задаче коллаборативной фильтрации каждый объект — *Оценка пользователя*, представлен двумя категориальными признаками: *Пользователь* и *Предмет*. По обучающей выборке оценок пользователей необходимо научиться предсказывать оценки для еще не известных пар (*Пользователь*, *Предмет*). Основным толчком для изучения подобных задач стало трехлетнее соревнование Netflix Prize [7].

Похожей является задача тематического моделирования. В ней объекты задаются двумя признаками *Документ*, *Термин* и частотами вхождения соответствующего термина в соответствующий документ [22]. Однако по обучающей выборке необходимо научиться не предсказывать частоты, а выделять тематики в документах и их вероятностные распределения. Поэтому о тематическом моделировании чаще говорят, как о задаче обучения без учителя.

В подобном виде также можно представить и задачу поискового ранжирования. В ней проводится предсказание релевантности для пары *(Запрос, Документ)*. В реальных поисковых системах этой паре соответствует большое количество признаков, посчитанных по ним, как вещественных, так и категориальных. Такое представление легко обобщается на различные интересные и крайне полезные для промышленности случаи. Например, предсказание релевантности для тройки *(Запрос, Документ, Пользователь)* будет соответствовать алгоритму персонализированного поиска.

3 Используемые методы

3.1 Группировка значений признаков

В настоящей работе часто будет использоваться прием группировки признаков. Он заключается в том, что фиксируется натуральное число $p \geq 1$ и генерируются новые перекодированные описания Z_i объектов X_i. Каждый признак Z^j также является категориальным и объединяет в себе информацию о некотором наборе из p признаков исходной матрицы данных. Каждому новому признаку соответствует некоторый набор из p исходных признаков. Кроме того, преобразование происходит таким образом, что

$$Z_{i'}^{j_1,\ldots,j_p} = Z_{i''}^{j_1,\ldots,j_p} \;\Leftrightarrow\; \forall\, l \in \{1,\ldots,p\} :\; X_{i'}^{j_l} = X_{i''}^{j_l}.$$

Таким образом, если изначально было m признаков, то их станет C_m^p. Подобный при-

ем иногда позволяет учитывать более глубокие взаимодействия между значениями признаков и, следовательно, может позволить алгоритмам достигать лучшего качества обучения. Однако следует заметить, что увеличение числа p несет существенное увеличение размерности пространства данных, что может сильно сказываться на эффективности работы методов по времени и памяти. Практически ко всем методам ниже можно применить группировку признаков [23].

3.2 Метод k ближайших соседей

В качестве базового решения был выбран метод k ближайших соседей (kNN). Алгоритм kNN требует способа вычисления расстояния между объектами — метрики. Многие известные метрики, например метрика Минковского, не позволяют учесть специфику категориальных значений. Поэтому была выбрана метрика Хемминга:

$$\rho(x_{i'}, x_{i''}) = \frac{1}{m} \sum_{j=1}^{m} I[X_{i'}^j \neq X_{i''}^j].$$

3.3 Случайный лес

Композиции решающих деревьев часто показывают высокое качество работы во многих задачах. Крайне популярным и эффективными методом является случайный лес (Random Forest) [12]. Построение каждого решающего дерева может учитывать и категориальные признаки. Для этого в каждом узле для выбора разбиения происходит перебор разбиений значений признаков на два множества, т.е. на левого и правого потомка в дереве. Поскольку перебор всевозможных разбиений значений признаков является очень вычислительно затратной задачей и имеет вычислительную сложность порядка $O(2^q)$, где q — количество принимаемых признаком значений, то на практике чаще используется вариант с перебором некоторой произвольной части разбиений.

3.4 Dummy-кодирование

Допустим, некоторый признак X^j принимает q значений $\{a_1, ..., a_q\}$. Тогда для каждого объекта X_i можно заменить признак X_i^j на q бинарных признаков следующим образом:

$$Z_i^{a_k} = I[X_i^j = a_k], \quad k \in \{1, ..., q\},$$

где $I[A]$ — индикатор события A, т.е.

$$I[A] = \begin{cases} 1, & \text{если } A \text{ истинно,} \\ 0, & \text{если } A \text{ не истинно.} \end{cases}$$

Такое кодирование называется dummy-кодированием или one-hot-кодированием. Если закодировать каждый признак исходной матрицы таким образом, то к полученным описаниям объектов $Z_1, ..., Z_n$ можно применять многие классические алгоритмы для работы с вещественными признаками.

Такой формат представления матрицы данных, несмотря на свои простоту и естественность, имеет несколько недостатков. Например, подобная перекодировка признаков накладывает ограничения на структуру признакового описания объектов, которую никак не учитывает алгоритм, работающий с вещественными признаками. Действительно, каждый признак перекодируется в несколько новых бинарных признаков с ровно одной единицей.

Еще одним недостатком такого подхода является сильно увеличивающаяся размерность пространства объектов. Многие алгоритмы не способны обрабатывать полученные матрицы данных во многих реальных задачах. В связи с этим описание объектов приходится хранить в разреженном формате и использовать приспособленные методы. Например, логистическую регрессию, многие реализации которой позволяют работать с разреженным представлением данных.

3.5 Произвольная перенумерация значений

Как уже было описано в части 2.2, простая нумерация значений признаков редко приводит к высокому результату из-за того, что алгоритмы начинают учитывать не имеющую смысла упорядоченность значений признаков. Такие алгоритмы получаются очень неустойчивыми относительно различных нумераций значений признаков и показывают низкое качество по отдельности. Однако, как показал Leo Breiman, подобные неустойчивые, но несмещенные алгоритмы можно крайне эффективно объединять в композиции, например с помощью техники bagging [11]. Таким образом можно обучать независимые слабые алгоритмы с произвольно пронумерованными признаками и в качестве финального предсказания брать усредненный ответ по всем ним. Подобный подход позволяет существенно повысить качество композиции по сравнению с каждым отдельным базовым алгоритмом.

3.6 Наивный байесовский классификатор

Для решения задачи бинарной классификации можно использовать мультиномиальный наивный байесовский классификатор. Его основная идея заключается в предположении о вероятностной условной независимости признаков. В случае классификации будет предсказан следующий класс:

$$\arg\max_y p(y|X) = \arg\max_y p(y) \prod_{j=1}^{m} p(X^j|y).$$

Для удобства дальнейших вычислений было проделано также и следующее преобразование:

$$\arg\max_y p(y) \prod_{j=1}^{m} \frac{p(y|X^j)p(X^j)}{p(y)} = \arg\max_y \frac{1}{p^{m-1}(y)} \prod_{j=1}^{m} p(y|X^j).$$

В случае задачи бинарной классификации выражение $\frac{1}{p^{m-1}(y)} \prod_{j=1}^{m} p(y|X^j)$ является «степенью принадлежности» объекта классу y. Если работать с метрикой AUC, то в качестве ответов алгоритмов можно использовать величину

$$\hat{y} = \frac{1}{p^{m-1}(y=1)} \prod_{j=1}^{m} p(y=1|X^j) \propto \prod_{j=1}^{m} p(y=1|X^j).$$

Поскольку все признаки принимают значения из дискретных неупорядоченных множеств, то по принципу максимума правдоподобия можно получить

$$p(y|X^j) = \frac{\sum_{i=1}^{n} I[y_i = y] \cdot I[X_i^j = X^j]}{\sum_{i=1}^{n} I[X_i^j = X^j]}.$$

В этой ситуации использование принципа максимума правдоподобия имеет несколько недостатков. Например, если в обучающей выборке не встречалось некоторого категориального значения признака j, то алгоритм не сможет вычислить $p(y|X_j)$. Кроме того, такой подход никак не учитывает дисперсию оценки максимума правдоподобия. На практике часто в подобных случаях используют аддитивное сглаживание вероятностей [17], накладывающее априорное распределение Дирихле на параметры вероятностной модели:

$$p(y|X^j) = \frac{\sum_{i=1}^{n} I[y_i = y] \cdot I[X_i^j = X^j] \; + \; \alpha \cdot \frac{\sum_{i=1}^{n} I[y_i = y]}{n}}{\sum_{i=1}^{n} I[X_i^j = X^j] + \alpha}, \quad \alpha \geq 0.$$

Параметр α является параметром сглаживания. Чем больше значения α — тем ближе оценки вероятностей к априорным вероятностям каждого из классов, а чем меньше α — тем ближе полученные оценки вероятностей к оценкам максимального правдоподобия.

На основе наивного байесовском классификатора можно строить и другие более подходящие для конкретных задач алгоритмы. Например, можно использовать его для задач регрессии [14].

Также можно использовать некий внешний мета-алгоритм f, который бы выражал искомую вероятность не простым произведением вероятностей, а каким-нибудь более сложным аппроксиматором:

$$\text{prediction} = f(p(y|x^1), ..., p(y|x^1)).$$

В качестве такого мета-алгоритма можно использовать любой классификатор или регрессор в зависимости от задачи. Например, если обучать линейный мета-алгоритм с весами $(w_1, ..., w_m)$ на логарифмах факторизованных вероятностей, то получим приближение

$$\text{prediction} = \prod_{j=1}^{m} p^{w_j}(y|x^j).$$

Подобный прием эффективен, однако его надо использовать корректно (см. секцию 3.8).

3.7 Аппроксимация целевых меток с помощью взвешенных разреженных разложений матриц

В задачах коллаборативной фильтрации классическим стал подход на основе взвешенного разложения разреженных матриц и тензоров. «Взвешенность» означает, что под разреженностью подразумеваются пропущенные значения, а не нулевые.

Рассмотрим некоторую пару признаков j_1 и j_2, которые принимают уникальные значения $\{a_k\}_{k=1}^{q_1}$ и $\{b_l\}_{l=1}^{q_2}$, соответственно. Тогда можно составить матрицу F размера $q_1 \times q_2$ оценок значения метки для соответствующей значений пары признаков j_1 и j_2. Если в обучающей выборке не встречались прецеденты для некоторой пары значений, то соответствующее значение матрицы считается неизвестным. Оценкой значения метки может служить, например, эмпирическое математическое ожидание значения метки:

$$F_{k,l} = \frac{\sum_{i=1}^{n} I[x_i^{j_1} = a_k] \cdot I[x_i^{j_2} = b_l] \cdot y_i}{\sum_{i=1}^{n} I[x_i^{j_1} = a_k] \cdot I[x_i^{j_2} = b_l]}.$$

К полученной матрице можно применять низкоранговое разреженное разложение ранга r:

$$F \approx GH, \quad G \in \mathbb{R}^{q_1 \times r}, \quad H \in \mathbb{R}^{r \times q_2},$$

где G и H — плотные (неразреженные матрицы). Таким образом любое, в том числе неизвестное значение F_{ij} можно приближенно восстановить по разреженному разложению. Далее над полученными значениями можно обучать внешний мета-алгоритм в соответствии с описанным в секции 3.8.

Назовем латентным вектором значения a_k признака j_1 k-ю строчку матрицы G и латентным вектором значения b_l признака j_2 l-й столбец матрицы H [18]. Вместо перекодировки исходных значений пар признаков восстановленными с помощью матричного разложения значений целевых меток можно использовать перекодировку соответствующими латентными векторами, полученными из матриц G и H. Таким образом каждый признак будет перекодирован новыми $2r$ признаками. Такая перекодировка может позволить внешним мета-алгоритмам находить более глубокие закономерности в данных.

Полностью аналогичные рассуждения можно проводить и для тензоров, т.е. рассматривать не пары признаков, а наборы признаков произвольной длины. В таком случае нужно использовать разреженные тензорные разложения.

При решении задач таким способом может возникнут сложность в том, что некоторые пары (a_k, b_l) могли ни разу не встречаться в обучающей выборке и, соответственно, невозможно будет провести перекодировку признаков на основе имеющегося разложения матрицы. В этом случае проделывалось следующее. Неизвестные значения (будь то a_k, или b_l) кодируются с помощью взвешенного среднего среди всех латентных векторов для значений, встретившихся в обучении, с весами, пропорциональными количествам вхождений соответствующего значения признака в обучение.

3.8 Стекинг алгоритмов

Техника использования внешнего мета-алгоритма над результатами работы других базовых алгоритмов известна давно и называется stacking [20].

Если обучать базовые алгоритмы и внешний мета-алгоритм на одной и той же выборке, то зачастую происходит эффект переобучения в силу того, что на вход внешнему мета-алгоритму даются перекодированные с помощью базовых алгоритмов признаки. Причем обучение перекодировок происходило на тех же данных, на которых и выполняется перекодировка.

В это случае необходимо разбивать обучающую выборку на две части. На одной части необходимо обучать базовые алгоритмы, кодирующие значения признаков, а на другой части необходимо обучать, собственно, сам внешний мета-алгоритм.

При использовании этого подхода возникают новые проблемы. Например, базовые алгоритмы и мета-алгоритм используют меньшие объемы данных для обучения и, следовательно, показывают более низкое качество, чем если бы обучались на всех данных. Также встает вопрос о выборке оптимальной пропорции разбиения обучающего множества на две части. Для такого разбиения можно использовать более сложные техники на основе кроссвадилации, однако они не рассматриваются в данной работе.

Стоит заметить, что на вход внешнему мета-алгоритму можно подавать не только от-

веты базовых классификаторов, а например, промежуточные вычисления в базовых алгоритмов. Это могут быть латентные векторы (см. раздел 3.7) или счетчики значений признаков, получающиеся в процессе работы алгоритмов на основе наивного байеса (см. раздел 3.6).

3.9 Использование частот совместных встречаемостей значений признаков для их перекодировки

3.9.1 Перекодировка частотами встречаемости наборов значений

Каждую пару значений признаков j_1 и j_2 можно перекодировать частотой совместной встречаемости:

$$z_i^{j_1,j_2} = \frac{1}{n}\sum_{k=1}^{n} I[x_i^{j_1} = x_k^{j_1}]I[x_i^{j_2} = x_k^{j_2}].$$

Такая перекодировка может позволить учесть генеративную природу данных.

Стоит обратить внимание, что подобная перекодировка не требует знания истинных меток объектов. Поэтому, если помимо обучающей выборки заранее известна и тестовая, для которой необходимо делать предсказания, как зачастую бывает в конкурсах по анализу данных, то можно использовать единую перекодировку как для обучения, так и для теста. Именно такой способ и использовался в экспериментах ниже. Если же тестовые данные заранее не известны, то перекодировку необходимо обучать лишь на обучающей выборке, поэтому на тестовой выборке может возникнуть проблема новых значений признаков. В этом случае можно использовать подход аналогичный описанному в разделе 3.7.

3.9.2 Разложение матриц частот

Помимо напрямую посчитанных оценок частот можно использовать и приближения частот, полученные в результате матричных разложений соответствующих матриц. Рассмотрим некоторую пару признаков j_1 и j_2, которые принимают уникальные значения $\{a_k\}_{k=1}^{q_1}$ и $\{b_l\}_{l=1}^{q_2}$, соответственно. Тогда можно составить матрицу F размера $q_1 \times q_2$. В этой матрице все значения известны — это частоты вхождения соответствующей пары значений признаков в обучающую выборку. Делая разложение матрицы

$$F \approx GH, \quad G \in \mathbb{R}^{q_1 \times r}, \quad H \in \mathbb{R}^{r \times q_2},$$

можно получить приближенные значения частот встречаемости соответствующих признаков. Поскольку многие значения частот равны нулю, то здесь можно также использовать разреженные матричные, но не в смысле пропущенных значений, а в смысле нулевых значений.

Вместо приближенных значений частот можно использовать перекодировку значений признаков соответствующими латентными векторами [24]:

$$Z_i^{j_1,j_2} = (G_{j_1,:}, H_{:,j_2}).$$

Такое кодирование будет содержать в себе информацию о совместной встречаемости значений признаков. Уже на матрице Z можно обучать мета-алгоритм. Стоит обратить внимание, что это подход без учителя и, соответственно, не требует дополнительного разбиения данных на несколько частей, как требовалось в некоторых подходах выше.

Возникает вопрос, что именно подразумевается под приближением одной матрицы P с помощью другой Q? В качестве метрики качества приближения можно использовать Фробениусову норму:

$$D_2(P, Q) = \| P - Q \|_F^2 = \sum_{i,j} (P_{i,j} - Q_{i,j})^2.$$

Однако куда более естественный смысл при аппроксимации частот имеет неотрицательное матричное разложение и обобщенная KL-дивергенция (или I-дивергенция) [13] в качестве меры качества разложения:

$$D_{\mathrm{KL}}(P \parallel Q) = \sum_{i,j} (P_{i,j} \cdot \log \frac{P_{i,j}}{Q_{i,j}} - P_{i,j} + Q_{i,j}).$$

Эти меры сходства являются частным случаем так называемой β-дивергенции [13]:

$$D_{\mathrm{Beta}}^{\beta}(P \parallel Q) = \sum_{i,j} (P_{i,j} \cdot \frac{P_{i,j}^{\beta-1} - Q_{i,j}^{\beta-1}}{\beta - 1} - \frac{P_{i,j}^{\beta} + Q_{i,j}^{\beta}}{\beta}).$$

Действительно, при $\beta \to 1$ имеем

$$D_{\mathrm{Beta}}^{\beta}(P \parallel Q) \to D_{\mathrm{KL}}(P \parallel Q),$$

а при $\beta = 2$

$$D_{\mathrm{Beta}}^{\beta}(P \parallel Q) = D_2(P, Q).$$

Стоит обратить внимание, что β-дивергенция, как в прочем и KL-дивергенция, в общем случае является несимметричной мерой сходства.

Аналогично можно рассматривать не пары признаков, а наборы признаков большей длины. В таком случае подход остается аналогичным, за исключением использования тензорного разложения вместо матричного. В настоящей работе подобные тензорные подходы не рассматривались.

4 Данные

Для экспериментов были выбраны два современных набора реальных данных с категориальными признаками. В данном разделе приведен краткий обзор этих наборов.

4.1 Amazon.com — Employee Access Challenge

4.1.1 Обзор набора данных

Первый набор данных был опубликован на международном соревновании по анализу данных *Amazon.com - Employee Access Challenge*, проводимом в 2013 году [6]. Далее в настоящей работе для краткости этот набор данных будет называться Amazon.

Необходимо решить задачу бинарной классификации. Всего в выборке 32769 объектов, из них классу 1 принадлежат $\approx 94\%$ объектов. Каждый объект описывается девятью категориальными признаками. Количество уникальных значений по каждому из признаков приведено в таблице 1.

Номер признака	1	2	3	4	5	6	7	8	9
Уникальных значений	7518	4243	128	177	449	343	2358	67	343

Таблица 1: Количество уникальных значений по каждому из признаков в наборе данных Amazon

4.1.2 Выбор способа тестирования алгоритмов

Для проведения всех экспериментов и более объективного сравнения результатов работы различных алгоритмов было решено зафиксировать единую конфигурацию тестирования для всех алгоритмов. Использование произвольных разбиений на обучение и контроль (в различных пропорциях) давало большее отклонение оценки качества, нежели кроссвадилация. На рис. 1 показан график зависимости оценки качества и ее 2σ-доверительный интервал в зависимости от количества фолдов в случае алгоритмов из принципиально разных семейств: логистической регрессии с dummy-кодированием и наивным байесовским классификаторов. В качестве σ^2 бралась эмпирическая несмещенная оценка дисперсии в каждой точке. В дальнейшем в работе везде будут использоваться несмещенные 2σ доверительные интервалы, если не обговорено обратное.

Видно, что при количестве фолдов, равном семи, оценка качества ведет себя уже довольно стабильно. Выбор большего числа фолдов влечет за собой большую вычислительную затратность. В связи с этим было решено использовать фиксированное разбиение на 7 фолдов.

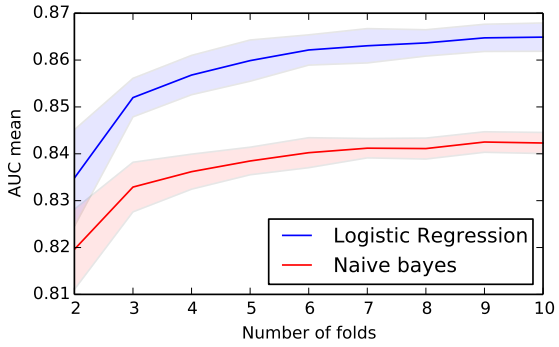


Рис. 1: Среднее качество предсказания в зависимости от количества фолдов и 2σ-доверительная трубка на данных Amazon

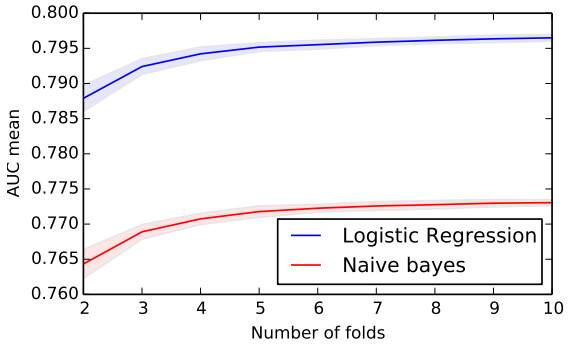


Рис. 2: Среднее качество предсказания в зависимости от количества фолдов и 2σ-доверительная трубка на данных Movie Lens

4.2 Movie Lens

4.2.1 Обзор набора данных

Второй набор данных был опубликован компанией Movie Lens [8] и называется Movie Lens 100K. В дальнейшем будем называть его просто Movie Lens.

Данные представляют из себя 100000 оценок пользователей некоторому набору кинофильмов. Каждый объект обучающей выборки — оценка пользователя. Категориальные признаки — номер пользователя, номер объекта, социальная информация о пользователе, категориальная информация о жанре соответствующего фильма. Изначально в вышеописанном наборе данных каждый из шанров был закодирован бинарным признаком, показывающим отношение фильма к соответствующему жанру (жанры могли пересекаться). Для дальнейших эсперимнтов все жанровые признаки были объединены в один жанровый признак со значениями, соответствующими каждой уникальной комбинации жанров. В качестве меток использовались бинарные метки, показывающие, была ли поставлена оценка 5.

Необходимо решить задачу бинарной классификации. Всего в выборке 100000 объектов, из них классу 1 принадлежат $\approx 64\%$ объектов. Каждый объект описывается шестью категориальными признаками. Количество уникальных значений по каждому из признаков приведено в таблице 2.

Номер признака	1	2	3	4	5	6
Уникальных значений	943	1682	2	21	795	216

Таблица 2: Количество уникальных значений по каждому из признаков в наборе данных Movie Lens

4.2.2 Выбор способа тестирования алгоритмов

Выбор конфигурации тестирования алгоритмов на этом наборе данных проводился аналогично подобной процедуре для набора Amazon. Графики зависимости и отклонения оценки качества в зависимости от количества фолдов показаны на рис. 2. Было решено использовать для всех экспериментов фиксированное разбиение на 5 фолдов.

5 Эксперименты

5.1 Используемая система для экспериментов

Все вычисления производились на компьютере Macbook Air выпуска середины 2012 года. Использовался процессор 2 ГГц Intel Core i7, 8 Гб оперативной памяти 1600 МГц DDR3, операционная система Mac OS X 10.9.2.

Практически все эксперименты были запрограммированы на языке Python 2.7.6 [9] с использованием библиотеки scikit-learn 14.1 [3]. Кроме того были использованы Matlab 2013a [1] и R 3.0.2 [10].

5.2 Метод ближайшего соседа

В качестве базового и самого простого алгоритма («бейзлайна») будем использовать метод k ближайших соседей с метрикой Хемминга. На рис. 3 показаны зависимости качества от количества ближайших соседей для обоих наборов данных.

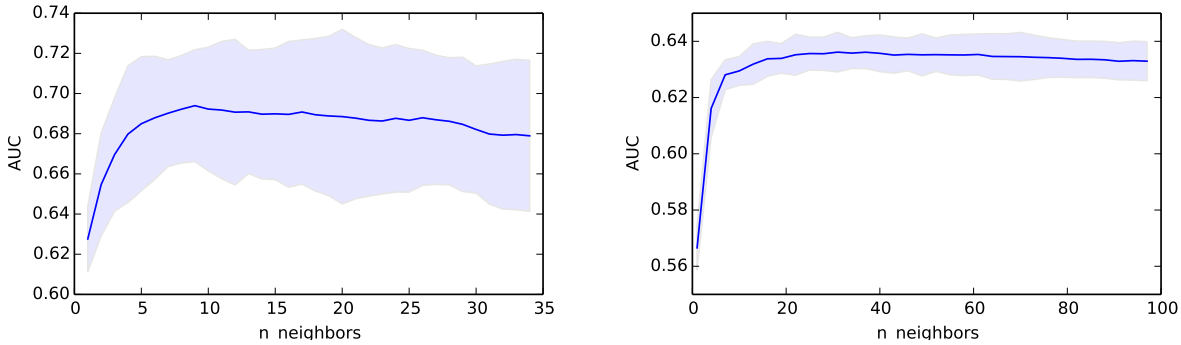


Рис. 3: Зависимость качества предсказания метода ближайших соседей от количества соседей. Слева изображен график для данных Amazon, справа — для данных Movie Lens.

Стоит заметить, что такой метод ближайших соседей показывает низкое качество не из-за принципиальных недостатков метода, а из-за наивного выборка метрики в пространстве объектов. При правильном выборе метрики, т.е. меры сходства между объектами, метод ближайших соседей может показывать очень высокие результаты, в том числе в этих задачах [24].

5.3 Наивный байесовский классификатор и его расширения

5.3.1 Классический наивный байес

В первую очередь были проведены эксперименты с обычным наивным байесовским классификатором. При аппроксимации вероятностей $p(y|X^j)$ значительную роль играет параметр аддитивного сглаживания α. На рис. 4 можно увидеть зависимость качества работы алгоритмов от параметра сглаживания. Графики приведены для оптимальных в соответствующей задаче значений степени группировки признаков p. Стоит заметить, что оптимальное значение α практически не меняется с увеличением p. Соответственно, можно подобрать α при фиксированном p и использовать его в дальнейших вычислениях.

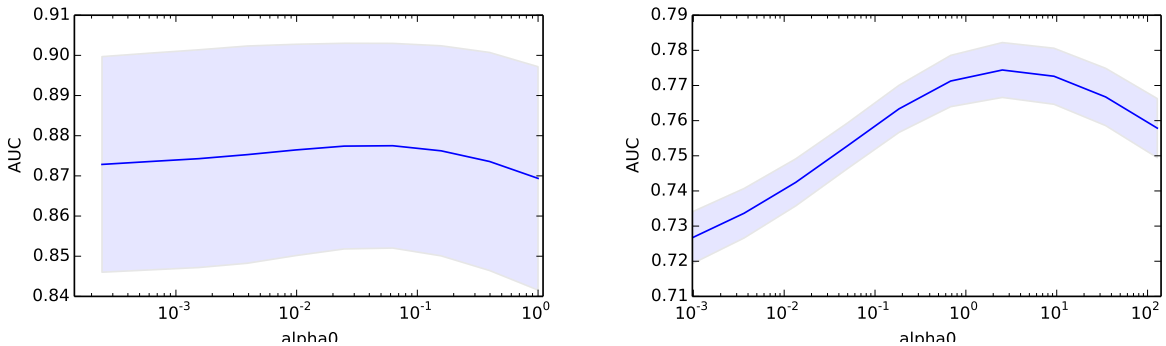


Рис. 4: Зависимость качества работы наивного байеса от параметра сглаживания. Слева изображен график для данных Amazon в случае группировки признаков по четверкам ($p = 4$), справа — для данных Movie Lens в случае группировки признаков попарно ($p = 2$).

Также существенно влияет на качество работы выбор параметра p — степени группировки значений признаков. На рис. 5 показана зависимость качества предсказания от степени группировки признаков. Видно, что на данных Amazon увеличение p повышает качество, а на данных Movie Lens оптимальным оказывается $p = 2$, а бóльшие значения существенно ухудшают результат.

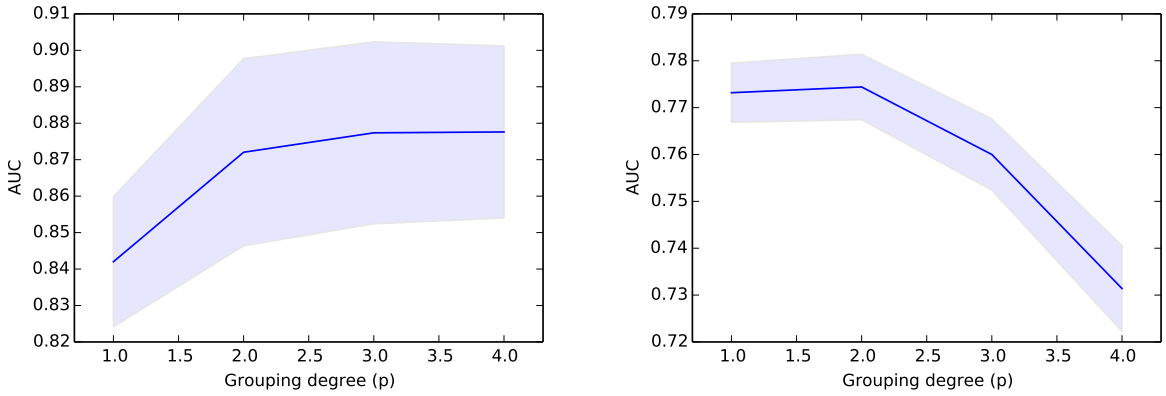


Рис. 5: Зависимость качества работы наивного байеса от степени группировки признаков (параметра p). Слева изображен график для данных Amazon, справа — для данных Movie Lens.

5.3.2 Обучение внешнего мета-алгоритма

Как было описано в разделе 3.8, можно использовать перекодировку значений признаков аппроксимациями соответствующих вероятностей $p(y|X^j)$ (а точнее, их логарифмов) и над полученной матрицей данных обучать мета-алгоритм. В качестве мета-алгоритмов были выбраны логистическая регрессия и случайный лес. В таблице 3 показана информация о результатах использования различных мета-алгоритмов. Видно, что логистическая регрессия является более предпочтительной как по качеству, так и по скорости работы.

Отдельно стоит заметить, что оптимальное p в случае классического наивного байесовского классификатора без использования внешнего мета-алгоритма не является оптимальным при его использовании. Так, например, $p = 4$ будучи оптимальным в случае классического наивного байеса на данных Amazon показывает низкое качество при использовании мета-алгоритма ($AUC = 0.8654$ против $AUC = 0.8721$).

Мета-алгоритм	**Amazon**	**Movie Lens**
Случайный лес	$AUC = 0.8658$	$AUC = 0.7655$
	время обучения = 17.8 сек	время обучения = 23.5 сек
	доля выборки = 40%	доля выборки = 30%
	$p = 2$	$p = 2$
Логистическая регрессия	$AUC = 0.8721$	$AUC = 0.7809$
	время обучения = 83.1 сек	время обучения = 76.4 сек
	доля выборки = 10%	доля выборки = 7%
	$p = 2$	$p = 2$

Таблица 3: Качество различных мета-алгоритмов, обучающихся над результатами работы наивного байеса

На рис. 6 показаны примеры зависимости качества от доли выборки, которая отходит на обучение мета-алгоритма.

Стоит заметить, что использование линейного мета-алгоритма требует значительно меньшей части обучающей выборки нежели случайный лес и, как следствие, перекодировка признаков обучается на большой доле данных. Возиожно, это является одной из причин того, что линейный мета-алгоритм показывает более высокое качество работы.

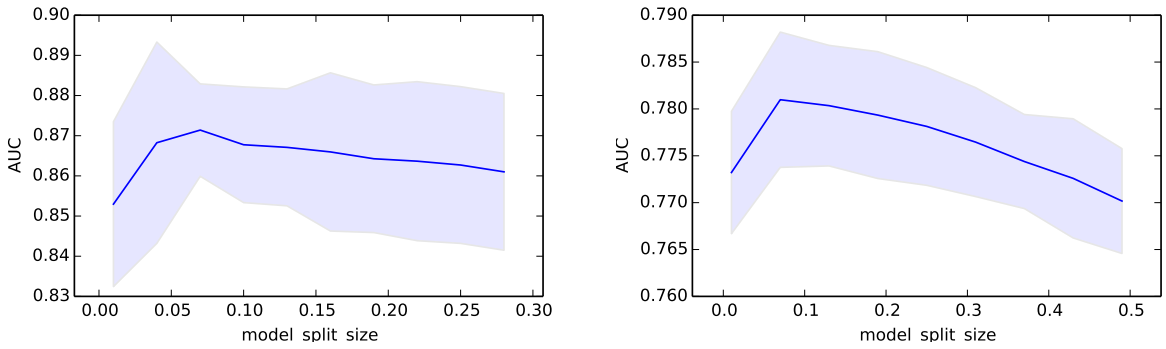


Рис. 6: Доля выборки, отходящая на обучение мета-классификатора. Слева изображен график для данных Amazon, справа — для данных Movie Lens.

Так же было проведено и множество других экспериментов. Например, были проведены попытки обучать мета-алгоритмы не на значениях $\log p(y|X^j)$, а на исходных $p(y|X^j)$. Однако прироста в качестве это не дало.

Также было замечено, что чаще всего неправильные ответы мета-алгоритм выдает на объектах, в которых накоплена маленькая частотная статистика для вычисления $p(y|X^j)$. В связи с этим была испробована идея использовать не

$$p(y|X^j) = \frac{\sum_{i=1}^{n} I[y_i = y] \cdot I[X_i^j = X^j]}{\sum_{i=1}^{n} I[X_i^j = X^j]}$$

в качестве признаков для мета-алгоритма, а отдельно числитель и знаменатель дроби, с помощью которой он вычисляется, чтобы мета-алгоритм смог улавливать уверенности и неуверенность частотных оценок. Однако и такой подход давал падение качества.

5.4 Разреженная логистическая регрессия

Одним и из наиболее успешных алгоритмов стала логистическая регрессия на данных, закодированных с помощью dummy-кодирования. Ключевыми параметрами здесь являются p (степень группировки признаков) и параметр регуляризации C логистической регрессии. На рис. 7 можно увидеть графики зависимости качества от параметра регуляризации при $p = 2$.

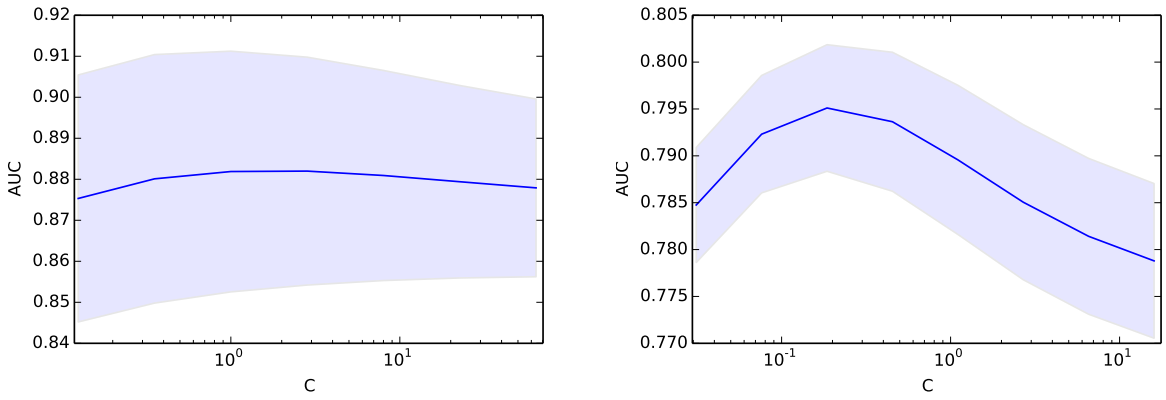


Рис. 7: Доля выборки, отходящая на обучение мета-классификатора

Вычисление при $p \geq 4$ требует больших вычислительных ресурсов по памяти. В таблице 4 показаны результаты работы при различных p. Видно, что группировка большой степени не имеет смысла и ведет к переобучению.

	Amazon	**Movie Lens**
$p = 1$	0.8691	**0.7958**
$p = 2$	**0.8820**	0.7951
$p = 3$	0.8802	0.7826

Таблица 4: Качество работы разреженной логистической регрессии при различных значениях степени группировки признаков p

Анализируя обученные линейные модели можно заметить, что большинство коэффициентов в ней близки к нулю. Этот эффект продемонстрирован на рис. 8. Это наводит на мысль об использовании модели с l_1-регуляризацией или elastic net [16], ведь они ведут к автоматическому отбору признаков и занулению коэффициентов. Однако такие подходы прироста качества не дали.

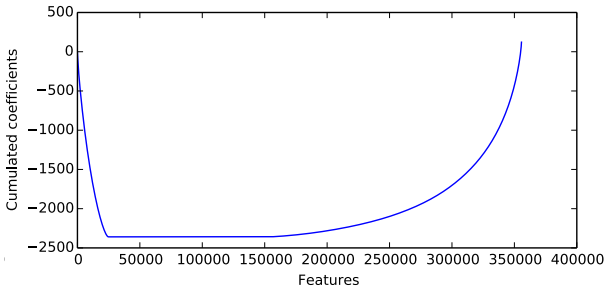


Рис. 8: Аккумулированные значения обученного и отсортированного вектора весов логистической регрессии на попарно сгруппированных данных Amazon

5.5 Произвольные перенумерации значений признаков

В этом разделе приведены результаты экспериментов с алгоритмами, описанными в разделе 3.5. Несмотря на простоту и банальность идеи, подобный подход показывает неплохое качество. На рис. 9 показана зависимость качества прогноза от количества базовых аппроксиматоров Extremely Randomized Trees [15] в случае сгруппированно попарно данных. Именно такие базовые аппроксиматоры показали себя лучше остальных опробованных.

Однако все же оно ниже чем у некоторых других подходов: $AUC = 0.8599$ на данных Amazon и $AUC = 0.7430$ на данных Movie Lens. Да и время работы оставляет желать лучшего. Например, на данных Amazon оно превосходит 5 минут, что сильно ограничивает возможности экспериментов с данным методом. Кроме того, оценка качества работы данного метода очень не устойчива.

Помимо обычного усреднения ответа по всем алгоритмам еще были проведены эксперименты с бустингом базовых алгоритмов, т.е. каждый следующий базовый алгоритм обучался так, чтобы восполнить ошибки предыдущих. Однако этот подход прироста не дал.

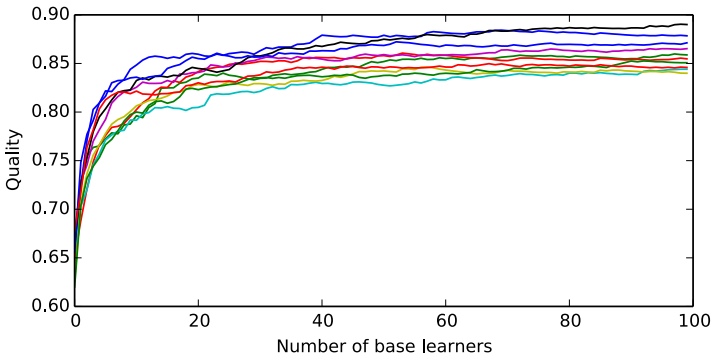


Рис. 9: Зависимость качества от количества базовых аппроксиматоров на попарно сгруппированных данных Amazon

5.6 Аппроксимация целевых меток с помощью матричных разложений

Были проведены эксперименты для алгоритмов, описанных в разделе 3.7. В качестве внешнего мета-алгоритма использовался случайный лес, он показал лучшее качество работы по сравнению с линейным классификатором. На наборе данных Amazon описанный метод достиг $AUC = 0.8726$, а на наборе Movie Lens — $AUC = 0.7816$.

К сожалению, более гибкая настройка этого семейства методов в данном работе не проводилась, хотя алгоритм позволяет имеет множество настроек. Например, можно пробовать варьировать пропорцию разбиения обучающей выборки для обучения ко-

дировки и мета-алгоритма, параметр аддитивного сглаживания α, использующийся при вычислении средних значений целевых меток в известных клетках раскладывающейся матрицы, выбор оптимального количества компонент в разложении и т.п.

Матричные разложения производились с помощью библиотеки Divisi2 [5] на Python для создания рекомедательных систем.

Результаты экспериментов показаны в таблице ниже.

Метод	**Amazon**	**Movie Lens**
Аппроксимации меток + LR	0.8032	0.7507
Аппроксимации меток + RF	0.8593	0.7539
Латентные векторы по меткам + LR	0.8697	0.7784
Латентные векторы по меткам + RF	**0.8726**	**0.7816**

5.7 Перекодировки частотами

5.7.1 Различные способы перекодировки с помощью частот

Как было описано в разделе 3.9, для кодировки значений признаков можно использовать частоты совместной встречаемости признаков. Кодировать их можно, собственно, частотами, аппроксимациями частот, полученными в результате матричных разложения или соответствующими латентными вектрами, получающимися в результате этих разложений. Перекодировав таким образом признаки, на полученных данных можно обучать внешний мета-алгоритм. Результаты экспериментов приведены в таблице 5.

	Amazon		**Movie Lens**	
Мета-алгоритм	**RF**	**LR**	**RF**	**LR**
Частоты	0.8503	$<$	0.7597	$<$
Аппроксимации частот	0.8472	$<$	0.7539	$<$
Латентные векторы	**0.8786**	0.8442	**0.7649**	0.7174

Таблица 5: Результаты работы различных кодировок частотами совместных встречаемостей признаков и различных мета-алгоритмов (RF — случайный лес, LR — логистическая регрессия)

Видно, что наиболее удачным оказывается кодирование с помощью латентных векторов и случайного леса в качестве мета-алгоритма. В дальнейшем будем использовать именно такую комбинацию.

5.7.2 Выбор меры матричной близости

Как было описано в разделе 3.9, существуют различные меры матричных близостей, на основе которых ведутся матричные разложения. На рис. 10 показана зависимость качества от параметра β.

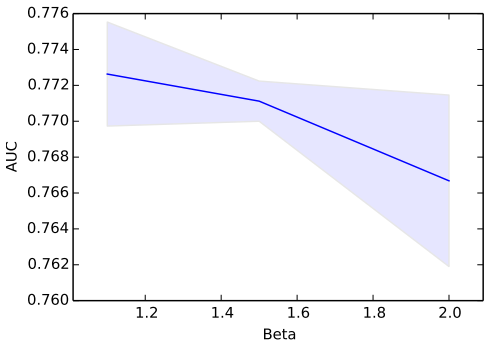


Рис. 10: Засимость качества аппроксимации от параметра β

Видно, что оптимизация среднеквадратичного отклонения (фробениусовой нормы D_2) не является оптимальным вариантом. Однако именно такая мера качества матричных аппроксимациях лучше всего изучена, обладает многими полезными свойствами и быстрыми прикладными реализациями.

Так, например, автор даже не смог дождаться выполнения разложения матриц на данных Amazon по β-дивергенции. Оно заняло существенно более 5 часов. Для оптимизации β-дивергенции в матричных разложениях использовалась библиотека [**?**].

Поэтому далее приводятся результаты именно для фробениусовой нормы. Для разложения матриц по ней использовалась библиотека [4]. Разложение любой пары признаков в обоих наборах данных происходило не более минуты.

5.7.3 Выбор количества компонент

Дополнительно были проведены эксперименты по определению оптимального количества компонент в таких разложениях. График зависимости качества от количества компонент на данных Amazon представлен на рис. 11. Видно, что увеличение количества компонент приводит к увеличению качества алгоритма. Переломный момент, когда большое количество компонент ведет к переобучению, обнаружен не был, однако он, скорее всего, есть. Стоит заметить, что алгоритму удается достигнуть высокого результата, что подтвержает оправданность и эффективность частотных кодировков признаков с помощью латентных векторов.

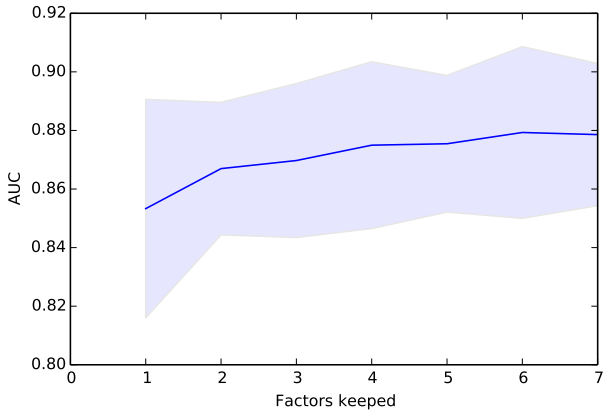


Рис. 11: Засимость качества от количества компонент

Помимо действий описанных выше, были проведены схожие эксперименты, в которых производились разложения матриц количеств совместных встречаемостей значений признаков и их видоизменений (например, логарифмов). Однако подобные подходы не смогли улучшить качество.

6 Заключение

6.1 Выводы

На данный момент не существует общепринятого стандартного «де-факто» набора алгоритмов для решения задач машинного обучения с категориальными признаками. В настоящей работе был проведен обзор существующих подходов, были представлены новые методы и сравнены эффективности их работы на реальных данных.

Стоит отметить, что в работе уделялось внимание алгоритмам по отдельности и не рассматривались объединения алгоритмов из разных семейств в композиции. Известно, что подобные техники позволяют сильно улучшать итоговое качество работы систем машинного обучения, даже если базовые алгоритмы показывали не очень высокое качество [24].

6.2 Что выносится на защиту

На защиту выносятся:

1. Обзор существующих эффективных подходов для решения задач с категориальными признаками;

2. Новые методы для решения задач с категориальными признаками;

3. Программная реализация описанных методов;

4. Эксперименты с предложенными методами на нескольких наборах реальных данных.

Список литературы

[1] Среда программирования matlab. http://www.mathworks.com.

[2] Библиотека matlab для неотрицательного матричного и тензорного разложения с оптимизацией β-дивергенции. http://www.mathworks.com/matlabcentral/fileexchange/38109-nonnegative-matrix-and-tensor-factorization--nmf--ntf--with-any-beta-diverg

[3] Библиотека scikit-learn для машинного обучения на python. http://scikit-learn.org.

[4] Библиотека scikit-tensor для python для разреженных тензорных разложений. https://github.com/mnick/scikit-tensor.

[5] Библитека divisi2 для python для создания рекомендательных систем на основе взвешенных матричных разложений. https://github.com/commonsense/divisi2.

[6] Международное соревнование amazon employee access challenge. http://www.kaggle.com/c/amazon-employee-access-challenge.

[7] Международное соревнование netflix prize. http://www.netflixprize.com.

[8] Набор данных movie lens 100k. http://grouplens.org/datasets/movielens.

[9] Язык программирования python. http://python.org.

[10] Язык программирования r. http://www.r-project.org.

[11] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[12] Leo Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.

[13] Andrzej Cichocki, Rafal Zdunek, Anh Huy Phan, and Shun ichi Amari. *Nonnegative Matrix and Tensor Factorizations - Applications to Exploratory Multi-way Data Analysis and Blind Source Separation.* Wiley, 2009.

[14] Eibe Frank, Leonard E. Trigg, Geoffrey Holmes, and Ian H. Witten. Naive bayes for regression (technical note). *Machine Learning*, 41(1):5–25, 2000.

[15] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine Learning*, 63(1):3–42, 2006.

[16] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, New York, 2009.

[17] D. Jurafsky and J.H. Martin. *Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition*. Prentice Hall series in artificial intelligence. Prentice Hall, 2000.

[18] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434. ACM, 2008.

[19] Steffen Rendle. Factorization machines with libfm. *ACM TIST*, 3(3):57, 2012.

[20] D. H. Wolpert. Stacked generalization. *Neural Networks*, 5:214–259, 1992.

[21] К. В. Воронцов. Математические методы обучения по прецедентам (теория обучения машин). *Москва*, 2011.

[22] К. В. Воронцов. Вероятностное тематическое моделирование. *Москва*, 2013.

[23] А. Г. Дьяконов. Теория систем эквивалентностей для описания алгебраических замыканий обобщенной модели вычисления оценок. *Журнал вычислительной математики и математической физики*, 50(2):388–400, 2010.

[24] А.Г. Дьяконов. Методы решения задач классификации с категориальными признаками. *Прикладная математика и информатика*, 46, 2014.