УДК 004

Поддержка выбора решений на основе антагонистических игровых моделей в условиях неопределённости с применением машинного обучения

Купреенков Сергей Дмитриевич
Филиал ФГБОУ ВО НИУ «Московский энергетический институт» в г. Смоленске
Студент

Аннотация
В статье решается проблема построения платёжной матрицы антагонистической игры в условиях неопределённости с помощью машинного обучения, а также представлен алгоритм для поддержки выбора решений.
Ключевые слова: машинное обучение, теория игр, принятие решений

Support for choice of solutions based on antagonistic game models under uncertainty with the use of machine learning

Kupreenkov Sergey Dmitrievich
Smolensk branch of Federal Autonomous Educational Institution (National Research University), Smolensk
Student

Abstract
The article solves the problem of building a payment matrix of an antagonistic game in conditions of uncertainty with the help of machine learning, and also presents an algorithm to support the choice of solutions.
Keywords: machine learning, theory of games, making decision

На практике встречаются задачи, когда у ЛПР возникает потребность в принятии такого решения, что его результат будет зависеть от других лиц и ЛПР просто наблюдает над развитием ситуации. В таком случае крайне затруднительно построить платёжную матрицу. Однако, если такие ситуации происходили много раз и известно большинство их параметров, то можно использовать алгоритмы машинного обучения для выявления закономерностей и прогнозирования результатов.

1. Постановка задачи

Представим задачу следующим образом. Игроки имеют общий набор стратегий, из которого происходит их выбор, следовательно, одинаковых стратегий с обеих сторон быть не может. Игрок А и игрок В совершают свои личные ходы один за другим. Причём игроку В будет известно о стратегии

игрока А. Следовательно игрок В может иметь преимущество над игроком А. Тогда возникает задача выбора такой стратегии игроком А, которая бы позволила снизить вероятность применения сильной стратегии игроком В. Неопределённость снимается с помощью формулирования гипотезы [1], на основе которой утверждается критерий выбора оптимальной стратегии для игрока А.

Из вышеописанного выводятся следующие частные задачи:

1) разработать метод оценки стратегий в условиях неопределённости с применением машинного обучения;
2) сформировать требования к антагонистической модели игры;
3) разработать алгоритм функционирования поддержки выбора решений в условиях неопределённости.

2. Метода оценки стратегий в условиях неопределённости

Предлагаемый метод оценки стратегий состоит из следующих этапов.

Этап 1. Сбор информации для формирования набора данных.

Этап 2. Анализ данных.

Этап 3. Работа с признаками.

Этап 4. Разбиение набора данных на набор для обучения и тестирования.

Этап 5. Обучение модели с учителем и оценка качества.

Этап 6. Формирование оценок для матрицы принятия решений.

Далее будут кратко рассмотрены этапы предлагаемого метода оценки стратегий.

2.1. Сбор информации для формирования набора данных

Одна из самых важных частей процесса машинного обучения это интерпретация данных, с которыми происходит работа, и применимость этих данных к задаче, которую требуется решить. Очевидно, что неэффективным решением представляется, выбор случайным образом алгоритма обучения и подстановка в него необработанных данных. Прежде чем приступить к построению модели, необходимо понять, что представляет собой рассматриваемый набор данных. Каждый алгоритм отличается с точки зрения типа обрабатываемых данных и вида решаемых задач. Создавая модель машинного обучения, следует получить ответ на следующие вопросы:

- На какой вопрос(ы) требуется найти ответ? Собранные данные способны ответить на этот вопрос?
- Как лучше всего сформулировать вопрос(ы) с точки зрения задач машинного обучения?
- Собрано ли достаточное количество данных, чтобы составить представление о задаче, которую требуется решить?
- Какие признаки извлечены и помогут ли они получить правильные прогнозы?
- Каким образом следует измерять эффективность решения задачи?

- Как решение, полученное с помощью машинного обучения, будет взаимодействовать с другими компонентами исследования или бизнес-продукта?

В более широком контексте, алгоритмы и методы машинного обучения являются лишь этапом более крупного процесса, призванного решить конкретную задачу, и поэтому необходимо всегда учитывать полную схему всего процесса [2].

В настоящее время существуют готовые наборы данных, которые можно найти в сети Интернет. Обычно это файлы текстовых форматов CSV (Comma-Separated Values) или JSON (JavaScript Object Notation). Такие форматы текстовой информации самые распространенные и удобны для использования в анализе данных. Также можно использовать API (Application Programming Interface) для сбора информации с внешних источников, если такая возможность присутствует.

2.2. Анализ данных

Анализ данных – это хороший способ обнаружить аномалии и какие-то особенности в исследуемом явлении или проблеме. Например, вполне возможно, что некоторые величины были измерены в дюймах, а не сантиметрах. В реальном мире нестыковки в данных и другие неожиданности очень распространены [2].

Первым делом смотрят на первую пару строк набора данных. Для этого можно использовать python-библиотеку Pandas.

Основными структурами данных в Pandas являются классы Series и DataFrame. Первый из них представляет собой одномерный индексированный массив данных некоторого фиксированного типа. Второй – это двухмерная структура данных, представляющая собой таблицу, каждый столбец которой содержит данные одного типа. Можно представлять её как словарь объектов типа Series.

Перед началом следует сделать допущение, что набор данных загружен в объект «df» класса DataFrame и применяемые методы вызывается у объекта.

Для просмотра первых строк набора данных используют метод head. В таблице 2.1 представлены первые пять строк набора данных об оттоке клиентов телеком-оператора.

Таблица 2.1 – Первые 5 записей набора данных об оттоке клиентов

	State	Account length	Area code	International plan	Voice mail plan	Number vmail messages	Total day minutes	Total day calls	Total day charge	Total eve minutes	Total eve calls	Total eve charge	Total night minutes	Total night calls	Total night charge	Total intl minutes	Total intl calls	Total intl charge	Customer service calls	Churn
0	KS	128	415	No	Yes	25	265.1	110	45.07	197.4	99	16.78	244.7	91	11.01	10.0	3	2.70	1	False
1	OH	107	415	No	Yes	26	161.6	123	27.47	195.5	103	16.62	254.4	103	11.45	13.7	3	3.70	1	False
2	NJ	137	415	No	No	0	243.4	114	41.38	121.2	110	10.30	162.6	104	7.32	12.2	5	3.29	0	False
3	OH	84	408	Yes	No	0	299.4	71	50.90	61.9	88	5.26	196.9	89	8.86	6.6	7	1.78	2	False
4	OK	75	415	Yes	No	0	166.7	113	28.34	148.3	122	12.61	186.9	121	8.41	10.1	3	2.73	3	False

Каждая строка представляет собой одного клиента – объект исследования, а столбцы – признаки объекта.

Далее смотрят на размер данных с помощью метода *shape*. Результатом его работы будет размерность исследуемого массива данных. В данном случае это матрица в 3333 строк и 20 столбцов.

Чтобы просмотреть информацию о всех признаках набора используют методом *info*. Рассмотрим результат выполнения, представленный на рисунке 2.2.

Из рисунка видно, что один признак логический bool, три признака имеют тип object и 16 признаков числовые. Также с помощью метода *info* удобно посмотреть на пробелы в данных, в нашем случае в каждом столбце по 3333 наблюдения, а значит пропуски отсутствуют. Если же они присутствуют, то их заполняют нулями, средними значениями или интерполяцией по соседним точкам.

Метод *describe* показывает основные статистические характеристики данных по каждому числовому признаку (типы int64 и float64): число непропущенных значений, среднее, стандартное отклонение, диапазон, медиану, 0.25 и 0.75 квартили.

Результат применения описанного метода представлен в таблице 2.2.

Для категориальных (тип object) и булевых (тип bool) признаков можно воспользоваться методом *value_counts*.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3333 entries, 0 to 3332
Data columns (total 20 columns):
State                     3333 non-null object
Account length            3333 non-null int64
Area code                 3333 non-null int64
International plan        3333 non-null object
Voice mail plan           3333 non-null object
Number vmail messages     3333 non-null int64
Total day minutes         3333 non-null float64
Total day calls           3333 non-null int64
Total day charge          3333 non-null float64
Total eve minutes         3333 non-null float64
Total eve calls           3333 non-null int64
Total eve charge          3333 non-null float64
Total night minutes       3333 non-null float64
Total night calls         3333 non-null int64
Total night charge        3333 non-null float64
Total intl minutes        3333 non-null float64
Total intl calls          3333 non-null int64
Total intl charge         3333 non-null float64
Customer service calls    3333 non-null int64
Churn                     3333 non-null bool
dtypes: bool(1), float64(8), int64(8), object(3)
memory usage: 498.1+ KB
None
```

Рисунок 2.2 – Результат выполнения метода info

Таблица 2.2 – Результат применения метода describe

	Account length	Area code	Number vmail messages	Total day minutes	Total day calls	Total day charge	Total eve minutes	Total eve calls	Total eve charge	Total night minutes	Total night calls	Total night charge	Total intl minutes	Total intl calls	Total intl charge	Customer service calls	Churn
count	3333.00	3333.00	3333.00	3333.00	3333.00	3333.00	3333.00	3333.00	3333.00	3333.00	3333.00	3333.00	3333.00	3333.00	3333.00	3333.00	3333.00
mean	101.06	437.18	8.10	179.78	100.44	30.56	200.98	100.11	17.08	200.87	100.11	9.04	10.24	4.48	2.76	1.56	0.14
std	39.82	42.37	13.69	54.47	20.07	9.26	50.71	19.92	4.31	50.57	19.57	2.28	2.79	2.46	0.75	1.32	0.35
min	1.00	408.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	23.20	33.00	1.04	0.00	0.00	0.00	0.00	0.00
25%	74.00	408.00	0.00	143.70	87.00	24.43	166.60	87.00	14.16	167.00	87.00	7.52	8.50	3.00	2.30	1.00	0.00
50%	101.00	415.00	0.00	179.40	101.00	30.50	201.40	100.00	17.12	201.20	100.00	9.05	10.30	4.00	2.78	1.00	0.00
75%	127.00	510.00	20.00	216.40	114.00	36.79	235.30	114.00	20.00	235.30	113.00	10.59	12.10	6.00	3.27	2.00	0.00
max	243.00	510.00	51.00	350.80	165.00	59.64	363.70	170.00	30.91	395.00	175.00	17.77	20.00	20.00	5.40	9.00	1.00

Посмотрим на распределение данных по целевой переменной — Churn:

- 0 – 2850;
- 1 – 483.

Получается, что 2850 из 3333 являются лояльными и значение переменной Churn у них равно нулю [3].

Таким образом, первичный анализ данных служит для исправления исходной информации, чтобы использовать скорректированный набор данных непосредственно в алгоритмах машинного обучения.

2.3. Работа с признаками

Чтобы уменьшить время вычислений и повысить качество моделей проводится удаление признаков, не имеющих никакой полезности.

Например, если признак содержит повторяющиеся значения, то это первый кандидат на выбывание. Из этого можно предположить, что низковариативные признаки хуже, чем высоковариативные. Таким образом, можно отсекать признаки, чья дисперсия ниже определенной границы.

Самый простой способ нахождения оптимального набора признаков это перебор. Из всех признаков берется какое-то подмножество и обучается модель на признаках подмножества. Потом берется следующее подмножество и так далее. В результате получаем подмножество, дающее наилучший результат [4].

Далее приводятся самые распространённые методы работы с признками.

- Прямое кодирование категориальных признаков.
- Кодирование категориальных признаков числами.
- Биннинг.
- Одномерные нелинейные преобразования.
- Одномерные статистики.
- Отбор признаков на основе модели.
- Итеративный отбор признаков.
- Применение экспертных знаний.

2.4. Разбиение набора данных на набор для обучения и тестирования

На основе собранного набора данных требуется построить модель машинного обучения. Но прежде, чем применить модель к новому набору, следует убедиться в том, что модель на самом деле работает и ее прогнозам можно доверять.

К сожалению, для оценки качества модели нельзя использовать данные, которые были взяты для построения модели. Это обусловлено тем, что модель просто запомнит весь обучающий набор и поэтому она всегда будет предсказывать правильный класс для любой точки данных в обучающем наборе. Это «запоминание» ничего не говорит об обобщающей способности модели (другими словами, неизвестно, будет ли эта модель так же хорошо работать на новых данных).

Для оценки эффективности модели предъявляются новые размеченные данные (размеченные данные, которые она не видела раньше). Обычно это делается путем разбиения собранных размеченных данных на две части. Одна часть данных используется для построения модели машинного обучения и называется обучающими данными (training data) или обучающим набором (training set). Остальные данные будут использованы для оценки качества модели, их называют тестовыми данными (test data), тестовым набором (test set) или контрольным набором (hold-out set).

В python-библиотеке scikit-learn есть функция train_test_split, которая перемешивает набор данных и разбивает его на две части. Эта функция отбирает в обучающий набор 75% строк данных с соответствующими метками. Оставшиеся 25% данных с метками объявляются тестовым набором. Вопрос о том, сколько данных отбирать в обучающий и тестовый наборы, является дискуссионным, однако использование тестового набора, содержащего 25% данных, является хорошим правилом [5].

2.5. Обучение модели с учителем и оценка качества

Есть две основные задачи машинного обучения с учителем: классификация (classification) и регрессия (regression).

Цель классификации состоит в том, чтобы спрогнозировать метку класса (class label), которая представляет собой выбор из заранее определенного списка возможных вариантов. Классификация иногда разделяется на бинарную классификацию (binary classification), которая является частным случаем разделения на два класса, и мультиклассовую классификацию (multiclass classification), когда в классификации участвует более двух классов. Бинарную классификацию можно представить как попытку ответить на поставленный вопрос в формате «да/нет».

Цель регрессии состоит в том, чтобы спрогнозировать непрерывное число или число с плавающей точкой (floating-point number), если использовать термины программирования, или вещественное число (real number), если говорить языком математических терминов. Прогнозирование годового дохода человека в зависимости от его образования, возраста и места

жительства является примером регрессионной задачи. Прогнозируемое значение дохода представляет собой сумму (amount) и может быть любым числом в заданном диапазоне. Другой пример регрессионной задачи – прогнозирование объема урожая зерна на ферме в зависимости от таких атрибутов, как объем предыдущего урожая, погода, и количество сотрудников, работающих на ферме. И снова объем урожая может быть любым числом.

Поставленная задача заключается в сравнении стратегий, чтобы можно было сказать какая из них повысит вероятность выигрыша, следовательно, решается задача бинарной классификации, заключающаяся в том, чтобы сказать какая стратегия относятся к положительному классу (выигрышу, например), а какая к отрицательному (проигрышу).

Далее перечислены распространённые алгоритмы классификации, подходящие для решения поставленной задачи.

- Метод k-ближайших соседе.
- Логистическая регрессия.
- Деревья решений.

В задаче бинарной классификации метки принадлежат множеству {0, 1}. Объекты с меткой 1 будем называть положительными, а с меткой 0 – отрицательными. Алгоритм при этом может возвращать произвольное число, которое с помощью порога переводится в бинарный ответ: $a(x) = [b(x) > t]$.

Наиболее очевидной мерой качества в задаче классификации является доля правильных ответов $\text{accuracy} = \frac{1}{\ell}\sum_{i=1}^{l}[a(x_i) = y_i]$.

Данная метрика, однако, имеет существенный недостаток. Если взять порог t меньше минимального значения прогноза b(x) на выборке или больше максимального значения, то доля правильных ответов будет равна доле положительных и отрицательных ответов соответственно. Таким образом, если в выборке 950 отрицательных и 50 положительных объектов, то при тривиальном пороге $t = \max_i b(x_i)$ получается доля правильных ответов 0.95. Это означает, что доля положительных ответов сама по себе не несет никакой информации о качестве работы алгоритма a(x), и вместе с ней следует анализировать соотношение классов в выборке. Также полезно вместе с долей правильных ответов вычислять базовую долю – долю правильных ответов алгоритма, всегда выдающего наиболее мощный класс.

При сравнении различных методов машинного обучения принято сообщать относительное уменьшение ошибки. Пусть два алгоритма a_1 и a_2 с долями правильных ответов r_1 и r_2 соответственно, причем $r_2 > r_1$. Относительным уменьшением ошибки алгоритма a_2 называется величина $\frac{(1-r_1)-(1-r_2)}{1-r_1}$. Если доля ошибок была улучшена с 20% до 10%, то относительное улучшение составляет 50%. Если доля ошибок была улучшена с 50% до 25%, то относительное улучшение также равно 50%, хотя данный прирост кажется более существенным. Если же доля ошибок была улучшена

с 0.1% до 0.01%, то относительное улучшение составляет 90%, что совершенно не соответствует здравому смыслу.

2.6. Формирование оценок для платёжной матрицы

После того как модель машинного обучения прошла обучение и настройку ей можно пользоваться для получения оценок стратегий, которыми заполняется платёжная матрица.

Модель принимает на вход стратегию игрока А и стратегию игрока В, а на выходе получается вероятность победы игрока А над игроком В.

3. Алгоритм поддержки выбора решений на основе антагонистических игровых моделей в условиях неопределённости с применением машинного обучения.

3.1. Антагонистическая модель игры

К модели антагонистической игры выдвигаются следующие требования, согласно поставленной задачи в п.1:

- набор стратегий для игрока А и игрока В един;
- игрок А и игрок В не могут использовать одну и ту же стратегию;
- главная диагональ платёжной матрицы не содержит оценок (нули);
- игрок В знает, как походил игрок А;
- главная диагональ (одинаковые стратегии) матрицы игнорируется применяемыми способами поиска оптимальных стратегий.

3.2. Описание алгоритма

Данный алгоритм предназначен для формирования антагонистической платёжной матрицы и рекомендации оптимальных стратегий в условиях неопределённости для поддержки решений, принимаемых ЛПР.

Для снятия неопределённости ЛПР выдвигается гипотезу, по которой формируется критерий отбора оптимальных стратегий.

Перед использованием алгоритма подразумевается, что уже получена модель для оценки стратегий методом, описанным во втором пункте. То есть эта модель является функцией выигрыша игрока.

Далее представлена схема (рис 3.1) дальнейших действий по построению матрицы игры и рекомендаций оптимальных стратегий игроку.

Последовательность шагов алгоритма:

1) получение оценочной модели методом из второй главы;
2) выдвижение гипотезы о снятии неопределённости и формирование критерия отбора оптимальных стратегий;
3) формирование матрицы игры с помощью оценочной модели;
4) поиск оптимальных стратегий с помощью критерия из п.2;
5) переход к пункту 4 при поступлении нового запроса на поиск.

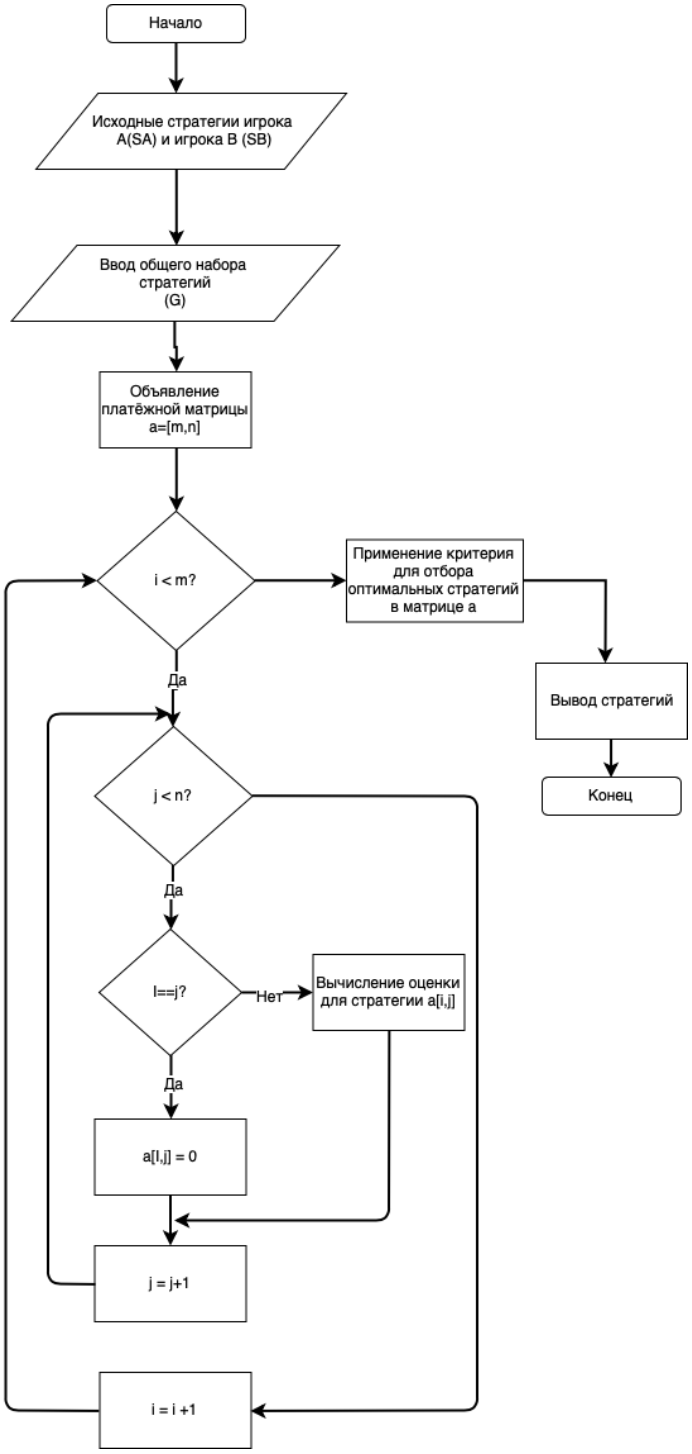

Рисунок 3.1 – Схема алгоритма построение матрицы игры

3.3. Применение алгоритма

Рассмотрим полученный алгоритм на примере в таблице 3.1. Пусть имеется некая модель, которая позволяет оценить шанс выигрыша игрока А при применении им стратегии A_i, когда игрок В отвечает стратегией B_j. Причем игроки выбирают стратегии из одного набора, в следствии чего, ситуация, в которой игроки выбирают друг против друга одну и ту же стратегию, недопустима.

Предполагается следующая гипотеза. Игрок А заинтересован в применении такой стратегии A_i, чтобы не дать игроку В сильного преимущества в выборе контрстратегии против себя. Поэтому он утверждает порог $a = 0{,}4$, который позволит для каждой стратегии A_i игрока А подсчитать количество стратегий B_j игрока В, в которой оценка a_{ij} больше порога a. Та строка матрицы, в которой будет больше всего таких стратегий, и будет считаться оптимальной для игрока А, так как снижается шанс применения сильной контрстратегии игроком В в силу его индивидуальных качеств.

Подсчитав стратегии, удовлетворяющие критерию, получается, что оптимальная стратегия это А2.

Таблица 3.1 – Матрица антагонистической игры

A\B	B1	B2	B3	B4	
A1	0	0,23	0,87	0,12	1
A2	**0,2**	**0**	**0,48**	**0,57**	**2**
A3	0.12	0.42	0	0,32	1

Заключение

В данной статье приведен метод для построения платёжной матрицы в условиях неопределённости с применением машинного обучения, что является сложной задачей в теории игр. Сформулированы требования, в которых разработанный алгоритм применяется. Приведен пример задачи, решаемый полученным алгоритмом.

Библиографический список

1. Луньков А. Д. Теория игр. Саратов: Саратовский государственный университет им. Н.Г. Чернышевскгого, 2008.
2. Мюллер А., Гвидо С. Введение в машинное обучение с помощью Python. Руководство для специалистов по работе с данными. СПБ.: ООО “Альфа-книга”, 2017
3. Открытый курс машинного обучения. Тема 1. Первичный анализ данных с Pandas. URL: https://habr.com/ru/company/ods/blog/322626/ (дата обращения 27.05.2019).
4. Открытый курс машинного обучения. Тема 6. Построение и отбор признаков. URL: https://habr.com/ru/company/ods/blog/325422/ (дата обращения 29.05.2019).
5. Мюллер А., Гвидо С. Указ соч. С. 31.