Практическая работа № 3. Методы машинного обучения для анализа данных

Работа № 3.1 Первичный анализ наборов данных

1. Источник данных: TurkeyStudentEvaluation

http://archive.ics.uci.edu/ml/datasets/Turkiye+Student+Evaluation

Свойства набора данных:

- Набор данных содержит ответы студентов на вопросы о качестве преподавания предметов
- Каждый вопрос оценивается баллами от 1 до 5
- 28 вопросов о качестве преподавания по пройденному предмету
- 3 преподавателя, 13 предметов
- 5820 объектов (записей)

Фрагмент источника данных

instr	class	nb.repeat	attendance	difficulty	Q1	Q2			Q19	Q20
1	2	1	3	5	3	3			3	3
1	2	1	3	4	3	3			3	3
1	2	1	0	1	5	5			5	5
1	2	1	3	5	3	3			3	3
1	2	1	3	4	5	5			5	5

Студенты оценивали параметры difficulty и Q1-Q28. Также приведена информация по посещаемости данным студентом некоторого занятия(attendance), какой раз данный студент проходит данный курс (nb.repeat), номер курса (class) и номер преподавателя (instr).

Описание атрибутов источника данных (на англ. из оригинала, на русском)

Attribute Information:

instr: Instructor's identifier; values taken from {1,2,3}
class: Course code (descriptor); values taken from {1-13}
repeat: Number of times the student is taking this course; values taken from {0,1,2,3,...}
attendance: Code of the level of attendance; values from {0, 1, 2, 3, 4}
difficulty: Level of difficulty of the course as perceived by the student; values taken from {1,2,3,4,5}
Q1: The semester course content, teaching method and evaluation system were provided at the start.
Q2: The course aims and objectives were clearly stated at the beginning of the period.
Q3: The course was worth the amount of credit assigned to it.
Q4: The course was taught according to the syllabus announced on the first day of class.
Q5: The class discussions, homework assignments, applications and studies were satisfactory.
Q6: The textbook and other courses resources were sufficient and up to date.
Q7: The course allowed field work, applications, laboratory, discussion and other studies.
Q8: The quizzes, assignments, projects and exams contributed to helping the learning.
Q9: I greatly enjoyed the class and was eager to actively participate during the lectures.
Q10: My initial expectations about the course were met at the end of the period or year.
Q11: The course was relevant and beneficial to my professional development.
Q12: The course helped me look at life and the world with a new perspective.
Q13: The Instructor's knowledge was relevant and up to date.
Q14: The Instructor came prepared for classes.

Q15: The Instructor taught in accordance with the announced lesson plan.
Q16: The Instructor was committed to the course and was understandable.
Q17: The Instructor arrived on time for classes.
Q18: The Instructor has a smooth and easy to follow delivery/speech.
Q19: The Instructor made effective use of class hours.
Q20: The Instructor explained the course and was eager to be helpful to students.
Q21: The Instructor demonstrated a positive approach to students.
Q22: The Instructor was open and respectful of the views of students about the course.
Q23: The Instructor encouraged participation in the course.
Q24: The Instructor gave relevant homework assignments/projects, and helped/guided students.
Q25: The Instructor responded to questions about the course inside and outside of the course.
Q26: The Instructor's evaluation system (midterm and final questions, projects, assignments, etc.) effectively measured the course objectives.
Q27: The Instructor provided solutions to exams and discussed them with students.
Q28: The Instructor treated all students in a right and objective manner.
Q1-Q28 are all Likert-type, meaning that the values are taken from {1,2,3,4,5}

Информация об полях источника данных:

instr: идентификатор инструктора; значения взяты из {1,2,3}
class: Код курса (дескриптор); значения взяты из {1-13}
repeat: сколько раз студент проходил этот курс; значения взяты из {0,1,2,3, ...}
attendance: Код уровня посещаемости; значения из {0, 1, 2, 3, 4}
difficulty: уровень сложности курса, который воспринимается студентом; значения взяты из {1,2,3,4,5}
Q1: Содержание семестрового курса, метод обучения и система оценивания были предоставлены в начале.
Q2: Цели и задачи курса были четко сформулированы в начале периода.
Q3: Курс стоил присвоенной ему суммы кредита.
Q4: Курс преподавался в соответствии с программой, объявленной в первый день занятий.
Q5: Обсуждения в классе, домашние задания, приложения и исследования были удовлетворительными.
Q6: Учебники и другие ресурсы курсов были достаточными и актуальными.
Q7: Курс допускал полевые работы, приложения, лабораторные, обсуждения и другие исследования.
Q8: Тесты, задания, проекты и экзамены способствовали обучению.
Q9: Мне очень понравился урок, и я очень хотел активно участвовать во время лекций.
Q10: Мои первоначальные ожидания относительно курса оправдались в конце периода или года.
Q11: Курс был актуален и полезен для моего профессионального развития.
Q12: Курс помог мне взглянуть на жизнь и мир с новой точки зрения.
Q13: Знания инструктора были актуальными и актуальными.
Q14: Инструктор прибыл подготовленным к занятиям.
Q15: Инструктор преподавал в соответствии с объявленным планом урока.
Q16: Инструктор был привержен курсу и был понятен.
Q17: Инструктор прибыл вовремя на занятия.
B18: Инструктор легко и четко произносит речь.
Q19: Инструктор эффективно использовал часы занятий.

Q20: Преподаватель объяснил курс и очень хотел помочь студентам.

Q21: Преподаватель продемонстрировал положительный подход к студентам.

Q22: Преподаватель был открыт и уважительно относился к мнению студентов о курсе.

Q23: Инструктор поощрял участие в курсе.

Q24: Преподаватель давал соответствующие домашние задания / проекты и помогал / руководил студентами.

B25: Инструктор ответил на вопросы о курсе внутри и вне курса.

Q26: Система оценки преподавателя (промежуточные и заключительные вопросы, проекты, задания и т. Д.) Эффективно измеряла цели курса.

Q27: Преподаватель предоставил решения к экзаменам и обсудил их со студентами.

Q28: Преподаватель относился ко всем студентам правильно и объективно.

Q1-Q28 все относятся к типу Лайкерта, что означает, что значения взяты из {1,2,3,4,5}

2. Описательные статистики

- Минимум и максимум
- Среднее значение
- Характеристики разброса
- Дисперсия
- Стандартное отклонение
- Интервал изменения
- Квантили
- Матрица ковариаций и корреляций (оценка связи между признаками)

Например: Вычисление описательных статистик NumPy:

```
import numpyas np
data = np.genfromtxt("turkiye.csv", delimiter=',')
for iin range(0,data.shape[1]):
# data.shape[1] размерность
    # второй мерности матрицы
    # (количество столбцов)
print("Признак ",i-2)
x = data[:,i]
    size = np.size(x)
    min = np.min(x)
    max = np.max(x)
print("Минимум: ",min)
print("Максимум: ",max)
    sum = np.sum(x)
print("Сумма: ",sum)
    sum2 = np.dot(x,x)
print("Суммаквадратов: ", sum2)
    mean = sum/size
    mean2 = sum2/size
print("Среднеезначение: ",mean)
print("Момент второго порядка: ", sum2/size)
var = np.var(x)
    SDM = var * size
print("Сумма квадратов отличий от средних: ", SDM)
print("Дисперсия: ",var)
    std = np.sqrt(var)
varcoef = std/mean
print("Среднеквадратичное отклонение: ",std)
print("Коэффициент вариации: ",varcoef)
print(" ")
corr = np.corrcoef(data, rowvar= 0)
print(np.max(corr))
```

```
print("Матрицакорреляции: ", corr)
covar = np.cov(data, rowvar= 0)
print("Матрицаковариаций: ", covar)
x = data[:,0]
print("Квантили: ",np.percentile(x,[25,50,75]))
```

Матрица корреляций признаков сложности предмета и всех вопросов.

	diff	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14	Q15	Q16	Q17	Q18	Q19	Q20	Q21	Q22	Q23	Q24	Q25	Q26	Q27	Q28
diff	1	0,05	0,07	0,07	0,06	0,06	0,05	0,05	0,05	0,06	0,04	0,06	0,04	0,08	0,09	0,09	0,05	0,12	0,07	0,08	0,09	0,1	0,1	0,08	0,07	0,1	0,06	0,06	0,09
Q1	0,05	1	0,87	0,77	0,85	0,8	0,77	0,79	0,79	0,73	0,8	0,72	0,76	0,72	0,7	0,7	0,74	0,61	0,71	0,7	0,69	0,67	0,67	0,73	0,73	0,67	0,7	0,71	0,66
Q2	0,07	0,87	1	0,85	0,87	0,86	0,83	0,84	0,83	0,8	0,85	0,79	0,8	0,8	0,79	0,79	0,81	0,72	0,79	0,79	0,78	0,76	0,77	0,8	0,8	0,77	0,78	0,77	0,75
Q3	0,07	0,77	0,85	1	0,83	0,84	0,82	0,82	0,81	0,8	0,83	0,81	0,78	0,81	0,81	0,8	0,79	0,77	0,8	0,8	0,8	0,79	0,79	0,8	0,79	0,79	0,8	0,77	0,78
Q4	0,06	0,85	0,87	0,83	1	0,87	0,84	0,84	0,82	0,78	0,84	0,77	0,79	0,78	0,77	0,78	0,79	0,7	0,77	0,77	0,76	0,75	0,75	0,79	0,79	0,75	0,77	0,76	0,74
Q5	0,06	0,8	0,86	0,84	0,87	1	0,88	0,89	0,88	0,81	0,88	0,81	0,82	0,83	0,81	0,81	0,84	0,73	0,82	0,81	0,79	0,78	0,78	0,83	0,83	0,78	0,8	0,79	0,77
Q6	0,05	0,77	0,83	0,82	0,84	0,88	1	0,89	0,86	0,8	0,87	0,8	0,81	0,81	0,8	0,8	0,82	0,72	0,78	0,79	0,78	0,77	0,77	0,8	0,8	0,77	0,79	0,78	0,76
Q7	0,05	0,79	0,84	0,82	0,84	0,89	0,89	1	0,9	0,82	0,89	0,81	0,83	0,81	0,79	0,79	0,82	0,7	0,79	0,79	0,78	0,76	0,76	0,82	0,82	0,77	0,8	0,79	0,75
Q8	0,05	0,79	0,83	0,81	0,82	0,88	0,86	0,9	1	0,83	0,89	0,81	0,84	0,79	0,78	0,77	0,82	0,7	0,79	0,78	0,77	0,75	0,75	0,81	0,82	0,76	0,79	0,79	0,73
Q9	0,06	0,73	0,8	0,8	0,78	0,81	0,8	0,82	0,83	1	0,87	0,83	0,81	0,79	0,79	0,79	0,8	0,75	0,79	0,79	0,78	0,78	0,78	0,79	0,78	0,78	0,78	0,76	0,76
Q10	0,04	0,8	0,85	0,83	0,84	0,88	0,87	0,89	0,89	0,87	1	0,86	0,87	0,84	0,82	0,82	0,86	0,73	0,83	0,82	0,81	0,8	0,8	0,84	0,84	0,79	0,83	0,82	0,78
Q11	0,06	0,72	0,79	0,81	0,77	0,81	0,8	0,81	0,81	0,83	0,86	1	0,86	0,8	0,8	0,8	0,79	0,75	0,78	0,8	0,79	0,79	0,79	0,79	0,79	0,78	0,78	0,77	0,77
Q12	0,04	0,76	0,8	0,78	0,79	0,82	0,81	0,83	0,84	0,81	0,87	0,86	1	0,79	0,77	0,76	0,8	0,69	0,78	0,78	0,76	0,75	0,74	0,79	0,8	0,74	0,77	0,77	0,73
Q13	0,08	0,72	0,8	0,81	0,78	0,83	0,81	0,81	0,79	0,79	0,84	0,8	0,79	1	0,94	0,91	0,9	0,84	0,89	0,88	0,88	0,87	0,87	0,87	0,86	0,87	0,86	0,83	0,86
Q14	0,09	0,7	0,79	0,81	0,77	0,81	0,8	0,79	0,78	0,79	0,82	0,8	0,77	0,94	1	0,93	0,89	0,88	0,89	0,89	0,9	0,89	0,89	0,87	0,86	0,89	0,86	0,83	0,87
Q15	0,09	0,7	0,79	0,8	0,78	0,81	0,8	0,79	0,77	0,79	0,82	0,8	0,76	0,91	0,93	1	0,89	0,88	0,89	0,89	0,89	0,89	0,89	0,88	0,85	0,89	0,86	0,82	0,87
Q16	0,05	0,74	0,81	0,79	0,79	0,84	0,82	0,82	0,82	0,8	0,86	0,79	0,8	0,9	0,89	0,89	1	0,8	0,91	0,88	0,87	0,85	0,85	0,89	0,88	0,85	0,86	0,85	0,83
Q17	0,12	0,61	0,72	0,77	0,7	0,73	0,72	0,7	0,7	0,75	0,73	0,75	0,69	0,84	0,88	0,88	0,8	1	0,85	0,86	0,87	0,87	0,87	0,82	0,79	0,87	0,82	0,77	0,86
Q18	0,07	0,71	0,79	0,8	0,77	0,82	0,78	0,79	0,79	0,79	0,83	0,78	0,78	0,89	0,89	0,89	0,91	0,85	1	0,9	0,88	0,87	0,87	0,88	0,87	0,86	0,86	0,83	0,84
Q19	0,08	0,7	0,79	0,8	0,77	0,81	0,79	0,79	0,78	0,79	0,82	0,8	0,78	0,88	0,89	0,89	0,88	0,86	0,9	1	0,91	0,9	0,89	0,89	0,87	0,88	0,87	0,84	0,86
Q20	0,09	0,69	0,78	0,8	0,76	0,79	0,78	0,78	0,77	0,78	0,81	0,79	0,76	0,88	0,9	0,89	0,87	0,87	0,88	0,91	1	0,93	0,91	0,89	0,86	0,89	0,87	0,83	0,88
Q21	0,1	0,67	0,76	0,79	0,75	0,78	0,77	0,76	0,75	0,78	0,8	0,79	0,75	0,87	0,89	0,89	0,85	0,87	0,87	0,9	0,93	1	0,94	0,89	0,86	0,9	0,87	0,84	0,89
Q22	0,1	0,67	0,77	0,79	0,75	0,78	0,77	0,76	0,75	0,78	0,8	0,79	0,74	0,87	0,89	0,89	0,85	0,87	0,87	0,89	0,91	0,94	1	0,9	0,87	0,91	0,87	0,84	0,89
Q23	0,08	0,73	0,8	0,8	0,79	0,83	0,8	0,82	0,81	0,79	0,84	0,79	0,79	0,87	0,87	0,88	0,89	0,82	0,88	0,89	0,89	0,89	0,9	1	0,92	0,89	0,88	0,87	0,86
Q24	0,07	0,73	0,8	0,79	0,79	0,83	0,8	0,82	0,82	0,78	0,84	0,79	0,8	0,86	0,86	0,85	0,88	0,79	0,87	0,87	0,86	0,86	0,87	0,92	1	0,88	0,88	0,87	0,84
Q25	0,1	0,67	0,77	0,79	0,75	0,78	0,77	0,77	0,76	0,78	0,79	0,78	0,74	0,87	0,89	0,89	0,85	0,87	0,86	0,88	0,89	0,9	0,91	0,89	0,88	1	0,89	0,85	0,9
Q26	0,06	0,7	0,78	0,8	0,77	0,8	0,79	0,8	0,79	0,78	0,83	0,78	0,77	0,86	0,86	0,86	0,86	0,82	0,86	0,87	0,87	0,87	0,87	0,88	0,88	0,89	1	0,88	0,88
Q27	0,06	0,71	0,77	0,77	0,76	0,79	0,78	0,79	0,79	0,76	0,82	0,77	0,77	0,83	0,83	0,82	0,85	0,77	0,83	0,84	0,83	0,84	0,84	0,87	0,87	0,85	0,88	1	0,85
Q28	0,09	0,66	0,75	0,78	0,74	0,77	0,76	0,75	0,73	0,76	0,78	0,77	0,73	0,86	0,87	0,87	0,83	0,86	0,84	0,86	0,88	0,89	0,89	0,86	0,84	0,9	0,88	0,85	1

Рисунок 3.1. Матрица корреляций признаков сложности предмета и всех вопросов

Видно, что вопросы, расположенные рядом обладают как правило большей корреляцией, чем вопросы, расположенные в опроснике дальше друг от друга. Это говорит о том, что опросник составлялся так, чтобы вопросы были расположены по некоторому логическому порядку, возможно группами, которые раскрывают одну из сторон преподавания. Также видно, что сложность фактически не коррелирует с ответами на вопросы.

3. Аномалии в данных

Причины появления аномалий в данных

- Неточности в данных, связанные с неточностью или ошибкой измерительных приборов, отказом оборудования
- Ошибки при сканировании, неточности, связанные с ошибкой распознавания
- Некорректная информация, полученная от людей - опрашиваемых, испытуемых.
- Ошибки при ручном создании наборов данных
  Поиск аномальных объектов
- Работа с пропущенными данными
- Избавление от несогласованности данных, подозрительно выделяющихся значений признаков, работа с выбросами
- Приведение числовых признаков к некоторому стандартному виду

Резюме: Аномалии в данных в разных наборах проявляются по-разному, выработать некоторый одинаковый подход сложно.

Поиск аномальных объектов

- Работа с пропущенными данными

- Избавление от несогласованности данных, подозрительно выделяющихся значений признаков, работа с выбросами
- Приведение числовых признаков к некоторому стандартному виду

Из-за наличия пропусков в данных работа некоторых алгоритмов невозможна. Пути решения – удаление признаков и объектов с большим количеством пропусков, подстановка средних значений по признаку, ЕМ алгоритм подстановки.

В разных местах одни и те же «связующие» данные могут быть записаны в разной форме. Некоторые значения не могут оказаться правдой в силу разных причин (возраст вряд ли может составлять >150, возраст, рост, вес не могут быть отрицательными и для них известно в каких пределах эти данные могут логично располагаться). Некоторые значения могут выбиваться из общего ряда просто потому, что случилось некоторое редкое, маловероятное событие, которое было записано.

Разные числовые признаки могут располагаться в разных диапазонах.

Рассмотрим пример работы с пропущенными данными. Можно заметить, что очень много пропусков у признака 3, объектов 3, 13, 15.

Объект\признак	1	2	3	4	5	Число пропусков	% пропусков
1	1.3	9.9	6.7	3.0	2.6	0	0
2	4.1	5.7			2.9	2	40
3		9.9		3.0		3	60
4	0.9	8.6		2.1	1.8	1	20
5	0.4	8.3		1.2	1.7	1	20
6	1.5	6.7	4.8		2.5	1	20
7	0.2	8.8	4.5	3.0	2.4	0	0
8	2.1	8.0	3.0	3.8	1.4	0	0
9	1.8	7.6		3.2	2.5	1	20
10	4.5	8.0		3.3	2.2	1	20
11	2.5	9.2		3.3	3.9	1	20
12	4.5	6.4	5.3	3.0	2.5	0	0
13					2.7	4	80
14	2.8	6.1	6.4		3.8	1	20
15	3.7			3.0		3	60
16	1.6	6.4	5.0		2.1	1	20
17	0.5	9.2		3.3	2.8	1	20
18	2.8	5.2	5.0		2.7	1	20
19	2.2	6.7		2.6	2.9	1	20
20	1.8	9.0	5.0	2.2	3.0	0	0
число пропусков	2	2	11	6	2	23	
% пропусков	10	10	55	30	10		23

После удаления указанных признаков и объектов остаётся лишь некоторое число пропусков, которые можно попытаться восстановить.

Объект\признак	1	2	4	5	число пропусков	% пропусков
1	1.3	9.9	3.0	2.6	0	0
2	4.1	5.7		2.9	1	25
4	0.9	8.6	2.1	1.8	0	0
5	0.4	8.3	1.2	1.7	0	0
6	1.5	6.7		2.5	1	25
7	0.2	8.8	3.0	2.4	0	0
8	2.1	8.0	3.8	1.4	0	0

Объект\признак	1	2	4	5	число пропусков	% пропусков
9	1.8	7.6	3.2	2.5	0	0
10	4.5	8.0	3.3	2.2	0	0
11	2.5	9.2	3.3	3.9	0	0
12	4.5	6.4	3.0	2.5	0	0
14	2.8	6.1		3.8	1	25
16	1.6	6.4		2.1	1	25
17	0.5	9.2	3.3	2.8	0	0
18	2.8	5.2		2.7	1	25
19	2.2	6.7	2.6	2.9	0	0
20	1.8	9.0	2.2	3.0	0	0
число пропусков	2	2	6	2	5	
% пропусков	0	0	29.4	0		7.35

Пример аномальных данных в TurkeyStudentEvaluation.

Можно заметить большое количество объектов, где все ответы на вопросы одинаковые. Есть некоторые, где все ответы на вопросы кроме оценки сложности одинаковые. В опросниках часто бывает такая ситуация, что человеку лень или нет времени отвечать на вопросы, поэтому он просто ставит одинаковые галочки на все вопросы. Возникает вопрос, какие из объектов действительно являются лишними в смысле их недостоверности. Например, 2 человека не посещали занятия, указали сложность предмета как 1 (очень легкий) и ответили на все вопросы о преподавателе 1 (что в целом говорит об оценке преподавателя как очень плохо): как студенты, не посещавшие занятия, могли оценить преподавателя как плохого во всём? Каждый может придумать свой критерий, какие объекты являются лишними.

Таблица Пример аномальных данных в TurkeyStudentEvaluation

instr	class	nb.repeat	attendance	difficulty	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10		Q11	Q12	Q13	Q14	Q15	Q16	Q17	Q18	Q19	Q20	Q21	Q22	Q23	Q24	Q25	Q26	Q27	Q28
1	2	1	3	5	3	3	3	3	3	3	3	3	3	3		3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
1	2	1	3	4	3	3	3	3	3	3	3	3	3	3		3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
1	2	1	0	1	5	5	5	5	5	5	5	5	5	5		5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
1	2	1	3	5	3	3	3	3	3	3	3	3	3	3		3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
1	2	1	3	4	5	5	5	5	5	5	5	5	5	5		5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
1	2	1	3	4	2	2	2	2	2	2	2	2	2	2		2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
1	2	2	1	5	3	3	2	2	5	3	3	3	5	5		4	4	3	4	4	4	4	4	4	4	4	5	4	5	5	4	4	4
1	2	1	2	4	1	1	4	2	3	3	2	2	2	2		3	2	4	3	3	3	5	2	3	3	3	1	3	3	1	3	3	2
1	7	3	0	4	3	3	3	3	3	3	3	3	3	3		3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
1	7	1	1	1	1	2	1	1	1	1	1	1	1	1		1	1	1	1	1	1	1	1	1	1	1	1	1	1	2	1	1	1
1	7	3	1	3	3	3	3	3	3	3	3	3	3	3		3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
1	7	1	0	1	1	1	1	1	1	1	1	1	1	1		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	7	1	0	1	1	1	1	1	1	1	1	1	1	1		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	7	2	4	5	5	5	5	5	5	5	5	5	5	5		5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
1	7	1	1	3	4	4	4	4	4	4	4	3	3	3		3	4	4	3	3	1	3	2	3	3	3	3	3	3	3	3	3	3
1	7	1	3	3	1	1	5	1	5	4	5	5	1	4		5	3	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
1	7	1	1	4	4	4	4	4	4	4	4	4	4	4		4	2	4	4	4	4	4	4	4	3	3	4	4	4	4	4	4	4
1	7	1	0	1	5	5	5	5	5	5	5	5	5	5		5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
1	7	1	2	2	3	3	4	4	4	3	4	5	3	3		4	3	4	3	4	3	4	3	4	3	2	3	3	3	4	4	4	3
1	7	1	4	4	4	5	5	4	5	4	4	5	5	5		5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
1	7	1	2	4	2	2	4	2	3	2	3	4	3	3		4	1	4	4	4	3	4	4	4	4	3	2	3	4	4	4	4	3
1	7	1	3	4	4	4	4	4	4	4	4	4	5	5		5	4	4	5	4	4	4	4	4	4	4	4	4	4	4	4	4	4

Методы поиска выбросов

- Поиск выбросов с использованием квартилей (данные выбросы ищутся по одному признаку):
  - Q_1 - значение признака, которое больше 25% значений из данных.
  - Q_3 - значение признака, которое больше 75% значений из данных
  - Выбросом является значение вне интервала $[X_1, X_2]$:

$$X_1 = Q_1 - k \cdot (Q_3 - Q_1) \qquad X_2 = Q_3 - k \cdot (Q_3 - Q_1)$$

- Поиск выбросов по распределениям признаков (данные выбросы могут быть найдены как многомерные – то есть целые объекты):
  - Все объекты, для которых выполнено неравенство, являются выбросами:

$$\sqrt{(x - \bar{x})' \sum_n^{-1} (x - \bar{x})} > g(n, \alpha_n)$$

где Σ – матрица ковариаций признаков.

Стандартизация и нормализация данных

- Стандартизация задаёт чёткие границы, в которых располагаются значения некоторого признака.

  Стандартизация: $a^j = \min_i x_i^j$, $b^j = \max_i x_i^j$

  1) $z_i^j = \frac{2x_i^j - (b^j + a^j)}{b^j - a^j}$, $z_i^j \in [-1,1]$

  2) $z_i^j = \frac{x_i^j - a^j}{b^j - a^j}$, $z_i^j \in [0,1]$
- Нормализация задаёт свойства признака – мат. ожидание 0 и стандартное отклонение 1.

  Нормализация: $\mu^j = \overline{x^j}$, $\sigma_j^2 = \frac{1}{n-1}\sum_i \left(x_i^j - \overline{x^j}\right)^2$

  $\overline{z^j} = 0$, $z_i^j = \frac{x_i^j - \mu^j}{\sigma^j}$, $\frac{1}{n-1}\sum_i \left(z_i^j - \overline{z^j}\right)^2 = 1$

Данные объекты были признаны выбросами (многомерными) среди первых 140 объектов (ответов студентов-посетителей курса №2). Можно заметить, что здесь присутствуют объекты с разными значениями признаков, причём существуют и более «крайние» объекты – из только единиц или пятерок – которые не попали в список выбросов. Применение метода поиска одномерных выбросов также не имеет смысла, поскольку зависит от значения квартилей. В некоторых случаях крайние оценки могут признаваться выбросом, если пользоваться этим методом, что не имеет смысла, поскольку если какая-то из оценок является выбросом, то нет никакого смысла использовать эту оценку при опросах.

Например, если k=1.5, $|Q_1-Q_3| = 2$, то можно показать, что все оценки будут признаны не выбросами. Если $|Q_1-Q_3| = 1$, то будет исключена как минимум 1 оценка из 5 (например, если $Q_1 = 2$, $Q_3 = 3$, то будет исключена оценка 5, если $Q_1 = 4$, $Q_3 = 5$, будут исключены 1 и 2). Если $|Q_1-Q_3| = 0$, будут исключены все оценки кроме $Q_1 = Q_3 =$ оценка.

Резюме:

- Ковариационная матрица близка к вырожденной (определитель ~0)
- Объекты в большинстве либо очень далеки от того, чтобы быть выбросами, либо выбросы при практически любом уровне значимости
- Объекты-выбросы практически не меняются при разумном изменении параметра уровня значимости
- Объекты, которые были сочтены выбросами не выглядят аномальными
- В данном случае анализ многомерных выбросов не имеет смысла. Необходимо придумать критерий удаления аномальных объектов.

Таблица. Пример объектов - выбросов базы TurkeyEvaluationStudent

i	c	nb	att	Diff	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
1	2	1	2	3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2	2	2	3	3	3	2	2	1	1	1	1
1	2	1	3	4	5	5	4	4	5	5	4	4	5	5	5	4	5	5	4	4	5	5	5	4	4	5	5	4	4	4	5	4
1	2	1	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	2	1	1	1	3	2	2	2	2
1	2	1	2	4	5	3	3	3	2	2	3	3	3	4	4	5	5	4	3	3	3	4	2	2	4	4	5	5	4	4	5	5
1	2	1	1	2	1	1	1	1	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
1	2	1	3	3	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	3	3	3	3	3	3	3
1	2	1	2	3	1	1	1	1	2	2	2	1	1	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
1	2	1	3	4	3	3	3	3	3	3	3	3	3	3	3	2	3	3	3	3	3	3	3	3	3	4	4	2	4	2	1	3
1	2	1	1	3	4	4	4	4	4	4	5	5	5	5	5	4	4	5	4	4	4	4	4	4	4	4	4	4	4	5	4	4
1	2	2	1	3	2	3	3	3	2	5	5	5	5	5	5	5	3	3	3	3	3	3	3	3	3	3	3	2	2	1	1	1
1	2	1	3	4	2	3	4	5	5	4	4	4	5	4	4	4	4	4	4	4	4	2	2	2	4	2	2	4	2	2	3	2
1	2	1	1	3	4	4	4	3	4	2	4	5	3	3	4	1	5	5	5	5	5	5	5	5	5	5	3	4	5	4	4	5
1	2	1	1	1	1	1	1	1	1	1	1	5	1	1	1	5	5	5	5	5	5	4	5	5	5	5	5	5	5	4	5	5
1	2	1	3	3	2	4	4	2	5	5	5	5	4	4	5	4	5	5	5	5	5	5	5	5	5	5	5	5	5	5	4	5
1	2	1	4	3	3	5	4	4	5	5	4	5	5	5	5	4	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
1	2	1	3	3	4	4	3	3	3	3	3	3	3	3	3	5	4	4	3	3	4	3	3	3	3	3	3	3	3	3	3	3
1	2	1	4	3	4	4	4	3	4	3	4	4	4	4	4	4	5	5	5	5	5	5	4	4	4	4	4	4	4	4	4	4
1	2	1	0	1	3	3	1	3	1	2	2	2	2	1	1	1	3	4	4	3	2	4	1	3	3	3	2	3	4	2	3	3
1	2	1	1	5	5	2	2	2	5	5	5	5	5	5	5	5	5	5	5	5	4	5	5	5	5	5	5	4	5	5	1	5
1	2	1	1	4	3	3	3	4	4	4	4	3	3	4	4	4	4	3	3	3	4	4	4	4	4	4	4	4	4	4	4	4
1	2	1	3	4	3	3	3	3	3	3	3	3	3	3	3	3	2	3	3	3	4	4	4	4	1	1	3	3	3	3	3	3

Практическое задание

1. Предложить методы анализа выбросов, учитывая особенности данных. Сделать анализ выбросов, удалить выбросы.
2. Проанализировать матрицу корреляций оценок по различным критериям качества преподавания. Выявить значимые корреляции. Объяснить высокие и низкие корреляции.
3. Сравнить матрицы корреляций для разных предметов.
4. Проанализировать описательные статистики по преподавателям, разработать метод сравнения преподавателей по приведённым данным.
5. Проанализировать описательные статистики по предметам, разработать метод сравнения предметов по данным из набора.

Контрольные вопросы:

- Как можно привести данные к единообразному виду?
- Какие есть инструменты для работы с данными?
- Какие простые метрики можно использовать для работы с данными?
- Как можно очистить данные от ненужных/мешающих элементов?
- Как работать с конкретными данными?
- Какие объекты можно признать аномальными в базе TurkeyStudentEvaluation?
- Какую информацию можно извлечь из данных?
- Как можно использовать эту информацию в будущем?

Работа № 3.2 Линейная регрессия

Цель работы: получение практических навыков построения и использования регрессионных моделей на языке Python с использованием библиотек Scikit-Learn и StatsModels.

Задание: используя программу Jupiter Notebook, язык программирования Python, библиотеки Scikit-Learn, StatsModels, NumPy, Matplotlib и др. выполнить следующие задания:

- **Парная регрессия**: построить две реализации парной линейной регрессионной модели на базе двух библиотек Scikit-Learn, StatsModels, сравнить и интерпретировать полученные результаты, входные данные рассчитать согласно варианту в таблице.
- **Множественная регрессия**: для своего варианта провести регрессионное моделирование (построить множественную регрессионную модель, ссылка для скачки данных на странице в разделе Data tables, выбрать не менее 50 строк):
  - выбрать выходную прогнозируемую переменную,
  - построить регрессионную модель со значимыми параметрами (оценить параметры межфакторной корреляции, последовательно добавлять факторы и сравнивать качество получаемых моделей, подобрать вид функции (визуальный анализ), оценить адекватность модели по статистическим показателям, каждый из этапов прокомментировать в отчете),
  - интерпретируете результаты моделирования (что значит полученная формула, какие переменные вносят больший вклад, что будет при изменении независимых переменных с зависимой),
  - прогнозировать новые значения с помощью построенной модели.

1. Теоретические сведения

Регрессия и классификация являются задачами машинного обучения с учителем. Обе используют сходную концепцию использования известных наборов данных, при этом используется алгоритм для изучения функции отображения входной переменной (x) в выходную переменную, то есть y=f(x). У задач есть общие черты и различия (Таблица).

Таблица – Сопоставление задач регрессии и классификации

	Регрессия	**Классификация**
Вид обучения	Обучение с учителем	
Задача	максимально точно аппроксимировать функцию отображения (f), чтобы при появлении новых входных данных (x) можно было прогнозировать выходные данные (y)	
Выходное значение	*числовые* или *непрерывные* (действительные числа)	*дискретные* или *категориальные*
Примеры алгоритмов	линейная регрессия, регрессия в алгоритмах опорных векторов SVR (support vector regression) и регрессионные деревья	логистическая регрессия, наивный байесовский алгоритм, решающие деревья и k-NN (k ближайших соседей)

Линейная регрессия (Linear regression) — один из самых фундаментальных алгоритмов, используемых для моделирования отношений между зависимой переменной и несколькими независимыми переменными. Целью обучения является поиск линии наилучшего соответствия.

Например: $\hat{y} = b_0 + b_1 x$ – уравнение линейной регрессии, которое описывает прямую, наиболее точно показывающую взаимосвязь между входными переменными **`x`** и выходными переменными **`y`**. Для составления этого уравнения нужно найти коэффициенты **`b`** для входных переменных.

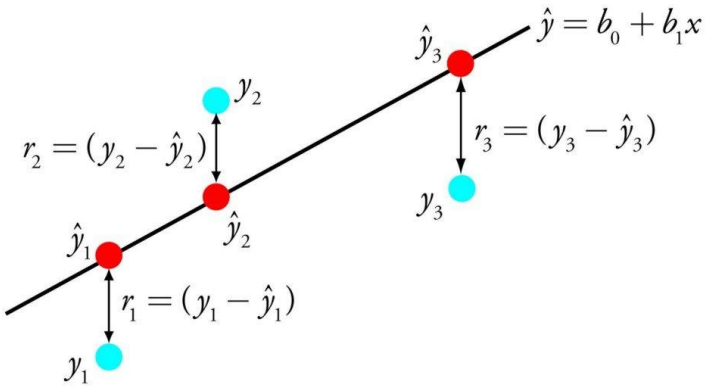


Рис. Графическое представление метода наименьших квадратов

В процессе обучения линейной регрессии находится минимизация квадратов расстояний между точками и линией наилучшего соответствия. Этот процесс известен как минимизация суммы квадратов остатков. Остаток равен разности между предсказанным значением и реальным.

$$J = \min_{b_i} \sum_{i=1}^{n} (y_i - \hat{y_i}))^2 = \min_{b_i} \sum_{i=1}^{n} (y_i - (b_0 + b_1 x_i))^2$$

Для оценки регрессионной модели используются различные методы вроде линейной алгебры или метода наименьших квадратов.

2. Задача

Рассмотрим задачу прогнозирования цен на рынке недвижимости.

- Какими данными о недвижимости мы можем располагать?
- Какие признаки влияют на цену?

Характеристики объектов недвижимости

1. Объективные характеристики:
   - *технический паспорт*
2. Субъективные характеристики (Как измерить?):
   - *состояние объекта недвижимости;*
   - *престижность района;*
   - *…*

Набор данных kc_house_data

https://www.kaggle.com/harlfoxem/housesalesprediction

Рассмотрим задачу прогнозирование цен на примере набора данных kc_house_data.

kc_house_data содержит данные о продажах индивидуальных домов в период с мая 2014 года по май 2015 в округе Кинг, штат Вашингтон, США.

Название признака	Описание
id	уникальный идентификационный номер проданного дома
date	дата продажи дома
bedrooms	количество спален
bathrooms	количество ванных комнат (где 0.25 обозначает, что комната с туалетом, 0.5 – комната с туалетом и раковиной)
sqft_living	общая площадь дома

sqft_lot	площадь прилегающей территории
floors	количество этажей
waterfront	бинарный атрибут, указывающий на то, есть ли вид на реку или нет
view	оценка внешнего вида дома (от 0 до 4)
condition	оценка состояния дома (от 0 до 5)
grade	оценка качества строительства и дизайна здания (от 1 до 13)
sqft_above	общая площадь наземной части дома
sqft_basement	общая площадь подземного части дома
yr_built	год строительства дома
yr_renovated	год последнего ремонта или последней реконструкции
zipcode	почтовый индекс дома
lat	широта
long	долгота
sqft_lot15	средняя общая площадь 15 ближайших домов
sqft_lot15	средняя площадь прилегающей территории 15 ближайших домов

3. Предобработка данных

Возможно ли уменьшить количество признаков?

Атрибуты *id*, *date, zipcode, lat, long.*

Удаляем *id*, *date, zipcode, lat, long* и sqft_basement (sqft_basement = sqft_living - sqft_above).

Пропуски в данных? Нет

Выбросы?

Формат данных: csv файл с разделителем в виде запятой. Используем библиотеку pandas.

```
import pandas as pd
from sklearn.model_selection import train_test_split

data = pd.read_csv("kc_house_data.csv", parse_dates= ['date']) # load the
data into a pandas dataframe
print(data.shape)
data.drop(['id', 'date','sqft_basement','lat','long'], axis = 1, inplace=
True)
print(data.shape)
for column in data:
print(column,end=';')
print()
# Формирование обучающей и тестовой выборок
train_data, test_data = train_test_split(data, train_size= 0.8,
random_state= 60)
train_data.astype(float).to_csv('kc_house_train_data.csv',sep=',',index=Fa
lse, header=False)
test_data.astype(float).to_csv('kc_house_test_data.csv',sep=',',index=Fals
e, header=False)
```

4. Модели регрессии

4.1. Линейная регрессия

Модель **линейной регрессии** имеет вид:

$price = w_0 + \sum_{j=1}^{m} w_j x^j + \varepsilon,$

где x^j - значение признака j.

Задача: найти коэффициенты w наиболее точно предсказывающие цены на имеющихся данных. Как измерить ошибку?

$$\varepsilon_i = price_i - w_0 - \sum_{j=1}^{m} w_j x_i^j$$

Определение **коэффициентов регрессии**

Введем обозначения:

$$\tilde{X} = (\tilde{x}_{ij}) = \begin{pmatrix} 1 & x_1^1 & \dots & x_1^m \\ \dots & \dots & \dots & \dots \\ 1 & x_n^1 & \dots & x_n^m \end{pmatrix}, w = \begin{pmatrix} w_0 \\ \dots \\ w_m \end{pmatrix}, y = \begin{pmatrix} y_1 \\ \dots \\ y_n \end{pmatrix}$$

В случае, когда матрица $\tilde{X}^T\tilde{X}$ невырождена, система нормальных уравнений имеет единственное решение:

$$\widehat{w} = (\tilde{X}^T\tilde{X})^{-1}\tilde{X}^T y$$

Каждый коэффициент регрессии отражает влияние конкретного признака на цену.

Если матрица $\tilde{X}^T\tilde{X}$ вырождена, система может не иметь решений или иметь бесконечное множество решений. Нет возможности интерпретировать коэффициенты регрессии.

Как быть, когда матрица вырождена или близка к вырожденной (плохо обусловленные или некорректные задачи)?

Значимые коэффициенты регрессии

Прогнозируемая переменная $Y = w_0 + \sum_{j=1}^{m} w_j x^j + \varepsilon,\ \varepsilon \sim \mathcal{N}(0, \sigma_\varepsilon)$, а коэффициенты регрессии, найденные с помощью МНК, равны $\widehat{w_0}, \dots, \widehat{w_m}$,. Как понять, что $w_j \neq 0$?

Z-score:

$$z_i = \frac{\widehat{w_i}}{\hat{\sigma}\sqrt{v_i}},\ \forall i = \overline{0, m}$$

где $\hat{\sigma} = \sqrt{\frac{1}{n-m-1}\sum_{i=1}^{n}(y_i - \widehat{w_0} - \sum_{j=1}^{m}\widehat{w_j}x^j)^2}$, v_i - диагональный элемент $(\tilde{X}^T\tilde{X})^{-1}$.

Если принять за нулевую гипотезу, что $w_j = 0$, то:

$$z_i \sim t_{n-m-1}$$

Чем больше абсолютное значение z_i, тем больше уверенность, что $w_j \neq 0$.

Модель линейной регрессия в Scikit-learn

```
from sklearn import linear_model

def linear_regression_model(dataX, dataY):
# Создать линейную регрессионную модель normalize=True
regr = linear_model.LinearRegression()
regr.fit(dataX, dataY) # Обучениемодели
return regr
```

Прогнозирование (Scikit-learn)

```
regressionModel.predict(data)
```

Коэффициент детерминации

Коэффициент детерминации. Коэффициент детерминации определяется следующим образом:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

где $\bar{y}$ - среднее значение по наблюдаемым данным, $\widehat{y_i}$ - значения объясняемой переменной, рассчитанные с помощью функции регрессии.

Коэффициент детерминации принимает значение от 0 до 1.

Для нахождения R^2 необходимо вызвать функцию r2_score

```
from sklearn.metrics import r2_score
r2_score (trueY, predictedY)
```

Среднеквадратичная ошибка

Для нахождения среднеквадратичной ошибки определим функцию *RMSE*.

Команда *mean_squared_error*, возвращает средний квадрат ошибки (MSE).

```
from sklearn.metrics import mean_squared_error
def RMSE(trueY, predictedY):
    return np.sqrt(mean_squared_error(trueY, predictedY))
```

Качество регрессии (kc_house_data)

- Как влияет формат данных на качество регрессии?
- Действительно ли стоимость дома линейно зависит от признаков bedrooms, bathrooms, floors, view, condition, grade?

Будем считать эти признаки категориальными и бинаризуем их.

```
categorial_cols = ['floors', 'view', 'condition', 'grade', 'bedrooms',
'bathrooms','zipcode']
for cc in categorial_cols:
    dummies = pd.get_dummies(data[cc], drop_first=False)
    dummies = dummies.add_prefix("{}#".format(cc))
data.drop(cc, axis=1, inplace=True)
    data = data.join(dummies)
print(data.shape)
for column in data:
print(column,end=';')
print()
```

В результате был получен набор с 79 признаками.

4.2. Гребневая регрессия

Гребневая регрессия также является линейной моделью регрессии, поэтому ее формула аналогична той, что используется в обычном методе наименьших квадратов. В гребневой регрессии коэффициенты (w) выбираются не только с точки зрения того, насколько хорошо они позволяют предсказывать на обучающих данных, они еще подгоняются в соответствии с дополнительным ограничением. Нам нужно, чтобы величина коэффициентов была как можно меньше. Другими словами, все элементы w должны быть близки к нулю. Это означает, что каждый признак должен иметь как можно меньшее влияние на результат (то есть каждый признак должен иметь небольшой регрессионный коэффициент) и в то же время он должен по-прежнему обладать хорошей прогнозной силой. Это ограничение является примером регуляризации (regularization). Регуляризация означает явное ограничение модели для предотвращения переобучения. Регуляризация, использующаяся в гребневой регрессии, известна как L2 регуляризация.

Метод гребневой регрессии решает проблему вырожденности с помощью добавления к функционалу Q регуляризатора, штрафующего большие значения квадрата евклидовой нормы вектора w:

$$Q_{\text{гр}}(w_0, \dots, w_m) := \left\|\tilde{X}w - y\right\|_2^2 + \alpha\|w\|_2^2 \to \min_w$$

где $\alpha \geq 0$

Для дифференцируемой функции $Q_{\text{гр}}$ необходимое условие минимума $\frac{\partial Q_{\text{гр}}}{\partial w_j} = 0, j = 0,1,\dots,m$ в матричной форме имеет вид:

$$2\tilde{X}^T\left(\tilde{X}\hat{w} - y\right) + 2\alpha I_n\hat{w} = 0 \Longrightarrow (\tilde{X}^T\tilde{X} + \alpha I_n)\hat{w} = \tilde{X}^T y$$

Откуда получаем, что, в случае невырожденности матрицы $\tilde{X}^T\tilde{X} + \alpha I_n$, решение системы нормальных уравнений выглядит следующим образом:

$$\hat{w} = (\tilde{X}^T\tilde{X} + \alpha I_n)^{-1}\tilde{X}^T y$$

Гребневая регрессии (Scikit-learn)

Гребневая регрессии реализована в классе linear_model.Ridge.

```
def ridge_regression_model(dataX, dataY, alphaParam):
# Инициализациямодели
ridge = linear_model.Ridge(alpha= alphaParam)
ridge.fit(dataX, dataY) # Обучение модели
return ridge
```

Модель Ridge позволяет найти компромисс между простотой модели (получением коэффициентов, близких к нулю) и качеством ее работы на обучающем наборе. Компромисс между простотой модели и качеством работы на обучающем наборе может быть задан пользователем при помощи параметра alpha. Оптимальное значение alpha зависит от конкретного используемого набора данных. Увеличение alpha заставляет коэффициенты сжиматься до близких к нулю значений, что снижает качество работы модели на обучающем наборе, но может улучшить ее обобщающую способность.

4.3. Лассо-регрессия

Альтернативой Ridge как метода регуляризации линейной регрессии является Lasso (англ.Least Absolute Shrinkage and Selection Operator, Оператор наименьшего сжатияи выбора). Как и гребневая регрессия, лассо также сжимает коэффициенты до близких к нулю значений, но несколько иным способом, называемым L1 регуляризацией. Результат L1 регуляризации заключается в том, что при использовании Лассо некоторые коэффициенты становятся равны точно нулю. Получается, что некоторые признаки полностью исключаются из модели. Это можно рассматривать как один из видов автоматического отбора признаков. Получение нулевых значений для некоторых коэффициентов часто упрощает интерпретацию модели и может выявить наиболее важные признаки вашей модели.

Метод Лассо штрафует большие значения L_1- нормы вектора w:

$$Q_{\text{л}}(w_0,\dots,w_m) \coloneqq \left\|\tilde{X}w - y\right\|_2^2 + \beta\|w\|_1 \to \min_w$$

где $\beta \geq 0$

Свойство:

При увеличении β количество коэффициентов регрессии равных нулю увеличивается, то есть происходит отбор значимых признаков (featureselection).

Функция $Q_{\text{л}}$ не является дифференцируемой функцией на всем множестве $\mathbb{R}^{m+1}$, но является субдифференцируемой, поэтому воспользуемся необходимым условием минимума в терминах субдифференциала:

$$0 \in \partial_{w_j} Q_{\text{л}}, j = 0,1,\dots,m$$

Подробно распишем чему равен $\partial_{w_j} Q_{\text{л}}$:

$$\partial_{w_j} Q_л = \frac{\partial Q}{\partial w_j} + \beta \partial_{w_j} |w_j| = -2\rho_j + 2z_j w_j + \begin{cases} -\beta, \text{ при } w_j < 0 \\ [-\beta, \beta], \text{ при } w_j = 0 \\ \beta, \text{ при } w_j < 0 \end{cases}$$

$$= \begin{cases} 2z_j w_j - 2\rho_j - \beta, \text{ при } w_j < 0 \\ [-2\rho_j - \beta, -2\rho_j + \beta], \text{ при } w_j = 0 \\ 2z_j w_j - 2\rho_j + \beta, \text{ при } w_j < 0 \end{cases}$$

где $\rho_j = \sum_{i=1}^{n} \tilde{x}_{ij}\left(y_i - \sum_{k=0.k\neq j}^{m} w_k \tilde{x}_{ik}\right), z_j = \sum_{i=1}^{n} \left(\tilde{x}_{ij}\right)^2$

Получаем, что $\hat{w}_j = \begin{cases} (\rho_j + \beta/2)/z_j, \text{ при } \rho_j < -\beta/2 \\ 0, \text{ при } \rho_j \in [-\beta/2, \beta/2] \\ (\rho_j - \beta/2)/z_j, \text{ при } \rho_j > \beta/2 \end{cases}$

В Scikit-learn для нахождения коэффициентов регрессии методом Лассо используется следующий функционал качества:

$$\frac{1}{2n} \left\| \tilde{X} w - y \right\|_2^2 + \alpha \|w\|_1 \to \min_w$$

где$\alpha \geq 0$

Лассо (Scikit-*learn*)

```
def lasso_model(dataX, dataY, alphaParam):
# Инициализациямодели
lasso = linear_model.Lasso(alpha=alphaParam, max_iter=1000)
lasso.fit(dataX, dataY)  # Обучение модели
return lasso
```

На практике, когда стоит выбор между гребневой регрессией и лассо, предпочтение, как правило, отдается гребневой регрессии. Однако, если у вас есть большое количество признаков и есть основания считать, что лишь некоторые из них важны, Lasso может быть оптимальным выбором. Аналогично, если вам нужна легко интерпретируемая модель, Lasso поможет получить такую модель, так как она выберет лишь подмножество входных признаков.

5. Практическое задание

1. Выполните предобработку данных (preprocessing):
   а) Анализ и удаление выбросов
   б) Анализ и восстановление пропусков
   в) Стандартизация данных
   г) Обсудить возможность выделения характерных признаков (featureextraction).
2. Выполнить вычисления по модели многомерной линейной регрессии и провести анализ полученной модели:
   а) Оценить качество регрессии по коэффициенту детерминации
   б) Оценить ошибку RMSE
   в) Выделить значимые и незначимые коэффициенты регрессии

Качество регрессии

	RMSE	R^2
Линейная регрессия		
Гребневая регрессия($\alpha = 3$)		
Лассо ($\alpha = 120$)		

Коэффициенты регрессии

	Линейная регрессия	Гребневая регрессия ($\alpha = 3$)	Лассо ($\alpha = 120$)
sqft_living			
sqft_lot			
waterfront			
sqft_above			
yr_built			
yr_renovated			
sqft_living15			
sqft_lot15			
floors#1.0			
floors#1.5			
floors#2.0			
floors#2.5			
floors#3.0			
floors#3.5			

Работа № 3.3 Анализ главных компонент

1. Теоретические сведения

Это один из основных алгоритмов машинного обучения. Позволяет уменьшить размерность данных, потеряв наименьшее количество информации. Вычисление главных компонент сводится к вычислению собственных векторов и собственных значений ковариационной матрицы исходных данных или к сингулярному разложению матрицы данных.

Метод главных компонент (МГК, англ. principal component analysis, PCA) — это техника машинного обучения, которая используется для изучения взаимосвязей между наборами переменных. Другими словами, МГК изучает наборы переменных для того, чтобы определить базовую структуру этих переменных. МГК еще иногда называют факторным анализом.

Основные приложения

- Dimensionality reduction. Снижение размерности данных при сохранении всей или большей части информации

Метод главных компонент используется для преобразования набора данных с множеством параметров в новый набор данных с меньшим количеством параметров и каждый новый параметр этого набора данных — это линейная комбинация раннее существующих параметров. Эти преобразованные данные стремятся обосновать большую часть дисперсии оригинального набора данных с гораздо большей простотой.

- Feature extraction. Выявление и интерпретация скрытых признаков

Нередко в машинном обучении встречаются ситуации, когда данные собираются априори, и лишь затем возникает необходимость разделить некоторую выборку по известным классам. Как следствие часто может возникнуть ситуация, когда имеющийся набор признаков плохо подходит для эффективной классификации. По крайней мере, при первом приближении.

В такой ситуации можно строить композиции слабо работающих по отдельности методов, а можно начать с обогащения данных путём выявления скрытых зависимостей между признаками. И затем строить на основе найденных зависимостей новые наборы признаков, некоторые из которых могут потенциально дать существенный прирост качества классификации.

Различия линейной регрессии и МГК

Линейная регрессия определяет линию наилучшего соответствия через набор данных. Метод главных компонентов определяет несколько ортогональных линий наилучшего соответствия для набора данных.

2. Задача: проанализировать заемщиков банка на основе различных данных

Пример источника данных: GiveMeSomeCredit
https://www.kaggle.com/c/GiveMeSomeCredit

Variable Name	Description	Type
RevolvingUtilizationOfUnsecuredLines	Total balance on credit cards and personal lines of credit except real estate and no installment debt like car loans divided by the sum of credit limits Общий баланс по кредитным картам и личным кредитным линиям, за исключением долга по недвижимости	percentage

	и без рассрочки, например автокредитов, деленный на сумму кредитных лимитов	
Age	Age of borrower in years Возрастзаемщика в годах	integer
NumberOfTime30-59DaysPastDueNotWorse	Number of times borrower has been 30-59 days past due but no worse in the last 2 years. Количество просроченных платежей заемщика на 30-59 дней, но не больше чем за последние 2 года.	integer
DebtRatio	Monthly debt payments, alimony, living costs divided by monthy gross income Ежемесячные выплаты по долгу, алименты, расходы на жизнь, разделенные на ежемесячный валовой доход	percentage
MonthlyIncome	Monthly income Ежемесячный доход	real
NumberOfOpenCreditLinesAndLoans	Number of Open loans (installment like car loan or mortgage) and Lines of credit (e.g. credit cards) Количество открытых займов (рассрочка, например, автокредит или ипотека) и кредитных линий (например, кредитные карты)	integer
NumberOfTimes90DaysLate	Number of times borrower has been 90 days or more past due. Количество просроченных платежей заемщика на 90 дней или более.	integer
NumberRealEstateLoansOrLines	Number of mortgage and real estate loans including home equity lines of credit Количество ипотечных кредитов и ссуд на недвижимость, включая кредитные линии под залог собственного капитала	integer
NumberOfTime60-89DaysPastDueNotWorse	Number of times borrower has been 60-89 days past due but no worse in the last 2 years. Количество раз, когда заемщик просрочил платеж на 60-89 дней, но не больше чем за последние 2 года.	integer
NumberOfDependents	Number of dependents in family excluding themselves (spouse, children etc.) Количество иждивенцев в семье, исключая их самих (супруга, дети и т. д.)	integer

Пример: Give Me Some Credit

Revolving Utilization Of Unsecured Lines	age	Number Of Time 30-59 Days Past Due Not Worse	Debt Ratio	Monthly Income	Number Of Open Credit Lines And Loans	Number Of Times 90 Days Late	Number Real Estate Loans Or Lines	Number Of Time 60-89 Days Past Due Not Worse	Number Of Dependents
0.766126609	45	2	0.802982129	9120	13	0	6	0	2
0.957151019	40	0	0.121876201	2600	4	0	0	0	1
0.65818014	38	1	0.085113375	3042	2	1	0	0	0
0.233809776	30	0	0.036049682	3300	5	0	0	0	0
0.9072394	49	1	0.024925695	63588	7	0	1	0	0
0.213178682	74	0	0.375606969	3500	3	0	1	0	1
0.305682465	57	0	5710	NA	8	0	3	0	0
0.754463648	39	0	0.209940017	3500	8	0	0	0	0
0.116950644	27	0	46	NA	2	0	0	0	NA
0.189169052	57	0	0.606290901	23684	9	0	4	0	2
0.644225962	30	0	0.30947621	2500	5	0	0	0	0
0.01879812	51	0	0.53152876	6501	7	0	2	0	2
0.010351857	46	0	0.298354075	12454	13	0	2	0	2
0.964672555	40	3	0.382964747	13700	9	3	1	1	2

3. Задача снижения размерности

Представить набор данных меньшим числом признаков таким образом, чтобы потеря информации, содержащейся в оригинальных данных, была минимальной.

Принципы компонентного анализа

Данные заданы матрицей $X = (x_i^j)$ размерности $n \times m$, где $i = \overline{1,n}$ и $j = \overline{1,m}$, n – число наблюдений (объектов), m – число признаков.

Обозначим за $C\ (m \times m)$ матрицу ковариаций признаков матрицы X:

$$c_{ij} = \frac{\sum_{p=1}^{n} x_k^i x_k^j}{n} - \mu_i \mu_j, \forall i,j \in \{1 \dots m\},$$

$$\mu_i - среднее значение признака i, i \in \{1 \dots m\}$$

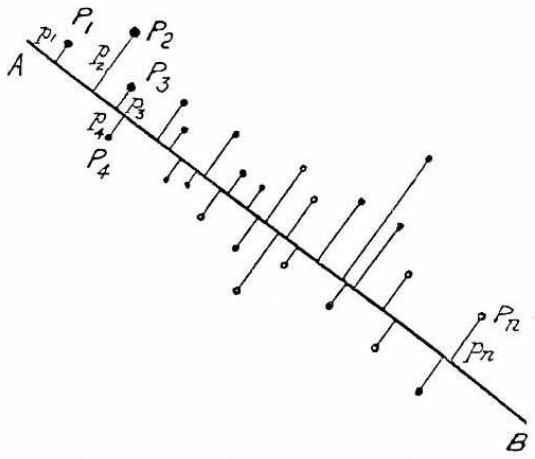
В матричном виде:

$$C = \frac{X^T X}{n} - \mu^T \mu, \mu = (\mu_1 \dots \mu_m)$$

Вариация i-го признака: $Var(x^i) = c_{ii}$

Общая вариация данных: $Var(X) = \sum_{i=1}^{m} c_{ii}$

Задача: найти ортогональные векторы такие, что $v^T C v \to max,$ т.е. проекция данных, на которые позволит сохранить наибольшую вариацию



Матрица C симметричная и положительно определена. Имеет место равенство:

$$C = V \Lambda V^T$$

$$\Lambda = \begin{bmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & \lambda_m \end{bmatrix},$$

$$\lambda - собственные значения матрицы C, \sum_{i=1}^{m} \lambda_i = \sum_{i=1}^{m} c_{ii}, \lambda_1 > \lambda_2 \geq \dots \geq \lambda_m \geq 0$$

$$V(m \times m) - матрица собственных векторов матрицы C$$

Главные компоненты:

$$U = X \cdot \lfloor v^1, v^2, \dots, v^k \rfloor^T, k < m$$

Доля объясненной вариации:

$$\frac{\sum_{i=1}^{k} \lambda_i}{Var(X)}$$

Singular value decomposition

- Данные заданы матрицей $X = (x_i^j)$ размерности $n \times m$, где $i = \overline{1,n}$ и $j = \overline{1,m}$, n – число наблюдений (объектов), m – число признаков.
- Требуется среди всех матриц такого же размера $n \times m$и ранга $\leq k$найти матрицу Y, для которой норма матрицы $\|X - Y\|$ будет минимальной.
- Решение зависит от матричной нормы
- Наиболее подходящие: Евклидова норма и норма Фробениуса:
    - Евклидова норма: $\|A\|_2 = \sqrt{\lambda T_{max}}$, где λ_{max} - максимальное собственное значение матрицы A
    - Норма Фробениуса: $\|A\|_F = \sqrt{\sum_i \sum_i a_{ij}^2}$

Существуют такие матрицы Uи V, что выполняется равенство $X = U \cdot S \cdot V^T$,

где U – матрица собственных векторов матрицы $X \cdot X^T$, V – матрица собственных векторов матрицы $X^T \cdot X$, а матрица S размерности $n \times m$имеет на главной диагонали элементы $\sigma_1, \sigma_2, \ldots, \sigma_m$ и все остальные нули, где σ_i - сингулярные числа матрицы X, а σ_i^2 - собственные числа матрицы $X^T \cdot X$.

Запишем матрицы Uи V в векторном виде:

$$U = \lfloor u^1, u^2, \ldots, u^n \rfloor, \rightleftarrows V = \lfloor v^1, v^2, \ldots, v^m \rfloor$$

Тогда SVD разложение можно представить как

$$X = \sigma_1 u^1 (v^1)^T + \sigma_2 u^2 (v^2)^T + \cdots + \sigma_m u^m (v^m)^T$$

Теорема Шмидта-Мирского:

Решением матричной задачи наилучшей аппроксимации в норме Евклида и в норме Фробениуса является матрица $X^* = \sigma_1 u^1 (v^1)^T + \sigma_2 u^2 (v^2)^T + \cdots + \sigma_k u^k (v^k)^T$

Ошибки аппроксимации:

$$\|X - X^*\|_2 = \sigma_{k+1}$$

$$\|X - X^*\|_F = \sqrt{\sigma_{k+1}^2 + \sigma_{k+2}^2 + \cdots \sigma_m^2}$$

Выбор числа k главных факторов

Общая вариация данных:

$$Var(X) = \sigma_1^2 + \sigma_2^2 + \ldots + \sigma_m^2$$

Доля объясненной вариации:

$$\frac{\sigma_1^2 + \sigma_2^2 + \ldots + \sigma_k^2}{Var(X)}, k < m$$

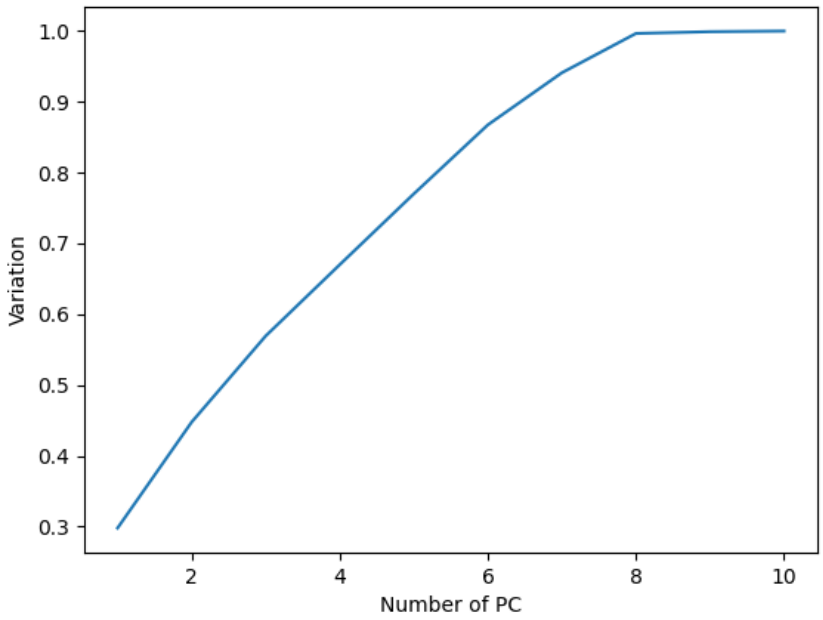
Хорошим значением считается доля объясненной вариации ≥ 80%

4. Решениевscikit-learn

```
import numpyas np
import scipyas sp
from sklearn.decompositionimport PCA
import matplotlib.pyplotas plt
from sklearn.preprocessingimport scale

np.set_printoptions(precision=10,
threshold= 10000,
suppress= True)
```

```
# Загружаем данные и удаляем наблюдения с пропущенными значениями
data = np.genfromtxt("cs-data.csv", delimiter = ',',
skip_header= 1, usecols=list(range(1, 11)))
data = data[~np.isnan(data).any(axis = 1)]

# Выполняем метод главных компонент
data = scale(data)
pca = PCA(svd_solver='full')
pca.fit(data)

print("Размерность данных \n", data.shape, "\n")
# Вклад каждого фактора в объяснение вариации
print("Вклад каждого фактора в объяснение вариации \n",
pca.explained_variance_ratio_, "\n")
# Рост доли объясненной вариации с увеличением числа главных факторов
var = np.round(np.cumsum(pca.explained_variance_ratio_), decimals=4)
print("Рост доли объясненной вариации с увеличением числа главных факторов
\n", var, "\n")
plt.figure()
plt.plot(np.arange(1,11), var)
plt.ylabel('Variation')
plt.xlabel('Number of PC')
plt.show()
```

```
Размерность данных
 (201669, 10)

Вклад каждого фактора в объяснение вариации
 [0.2979766397 0.1496007962 0.1217110055 0.1007219879 0.0999739517
 0.0975640598 0.0735527529 0.055468798  0.0024871325 0.0009428757]

Рост доли объясненной вариации с увеличением числа главных факторов
[0.298  0.4476 0.5693 0.67   0.77   0.8675 0.9411 0.9966 0.9991 1.    ]
```



5. Задание

1. Воспроизведите вычисления, представленные в теоретическом материале практической работы. Подтвердите выводы.

2. Рассмотрите набор данных Turkiye Student Evaluation:
    a) Опишите исследуемые данные
    b) Выберите данные по одному предмету (class) и выполните анализ главных компонент. Выделите главные факторы, дайте интерпретацию (или покажите, что этого сделать нельзя).
    c) Выберите два предмета, которые проводил один и тот же преподаватель. Снова выполните анализ главных компонент, выделите главные факторы, постарайтесь дать интерпретацию. Сравните результаты с предыдущим пунктом.
    d) Выполните PCA для всего набора данных. Также сравните результаты с пунктами выше.
    e) Повторите вычисления из пунктов b - d, но для нестандартизованных данных. Сравните с соответствующими результатами, полученными на стандартизованных данных.

Работа № 3.4 Наивный байесовский классификатор

1. Теоретические сведения

Метод классификации, основанный на наивном байесовском классификаторе, является алгоритмом обучения с учителем, в котором применяется теорема Байеса со строгим (наивным) предположением о независимости между каждыми парами признаков. Предположение о независимости позволяет избавиться от сложной схемы оценки параметров классификатора. Это позволяет применять алгоритм на больших выборках. Также классификация оказывается достаточно точной: недостаточной для высокоточных систем классификации, однако удовлетворительной для грубой оценки и сравнения с другими алгоритмами.

Вероятностная модель для классификатора – это условная модель $P(y \mid x_1, \dots, x_n)$ над независимой переменной класса y и признаками $x_1, \dots, x_n$ по теореме Байеса

$$P(y \mid x_1, \dots, x_n) = \frac{P(y)P(x_1, \dots, x_n \mid y)}{P(x_1, \dots, x_n)}$$

и предположении независимости, получаем:

$$P(x_i|y, x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n) = P(x_i|y),$$

Для всехi, это соотношение упрощается

$$P(y \mid x_1, \dots, x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \dots, x_n)}$$

Поскольку $P(x_1, \dots, x_n)$ является постоянной величиной с учетом входных данных, мы можем использовать следующее правило классификации:

$$P(y \mid x_1, \dots, x_n) \propto P(y)\prod_{i=1}^{n} P(x_i \mid y)$$

$$\Downarrow$$

$$\hat{y} = arg\max_{y} P(y)\prod_{i=1}^{n} P(x_i \mid y),$$

и использовать оценку апостериорного максимума для оценки $P(y)$ и $P(x_i \mid y)$. Различные наивные байесовские классификаторы различаются, в основном, допущениями, которые они делают относительно $P(x_i \mid y)$.

Причина, по которой наивные байесовские модели столь эффективны, заключается в том, что они оценивают параметры, рассматривая каждый признак отдельно и по каждому признаку собирают простые статистики классов.

Положительные и отрицательные стороны наивного байесовского алгоритма

Положительные стороны:

Классификация, в том числе многоклассовая, выполняется легко и быстро.

Когда допущение о независимости выполняется, НБА превосходит другие алгоритмы, такие как логистическая регрессия (logistic regression), и при этом требует меньший объем обучающих данных.

НБА лучше работает с категорийными признаками, чем с непрерывными. Для непрерывных признаков предполагается нормальное распределение, что является достаточно сильным допущением.

Отрицательные стороны:

Если в тестовом наборе данных присутствует некоторое значение категорийного признака, которое не встречалось в обучающем наборе данных, тогда модель присвоит нулевую вероятность этому значению и не сможет сделать прогноз. Это явление известно под названием «нулевая частота» (zerofrequency). Данную проблему можно решить с помощью сглаживания. Одним из самых простых методов является сглаживание по Лапласу (Laplacesmoothing).

Хотя НБА является хорошим классификатором, значения спрогнозированных вероятностей не всегда являются достаточно точными. Поэтому не следует слишком полагаться на результаты, возвращенные методом *predict_proba*.

Еще одним ограничением НБА является допущение о независимости признаков. В реальности наборы полностью независимых признаков встречаются крайне редко.

Приложения наивного байесовского алгоритма

Классификация в режиме реального времени. НБА очень быстро обучается, поэтому его можно использовать для обработки данных в режиме реального времени.

Многоклассовая классификация. НБА обеспечивает возможность многоклассовой классификации. Это позволяет прогнозировать вероятности для множества значений целевой переменной.

Классификация текстов, фильтрация спама, анализ тональности текста. При решении задач, связанных с классификацией текстов, НБА превосходит многие другие алгоритмы. Благодаря этому, данный алгоритм находит широкое применение в области фильтрации спама (идентификация спама в электронных письмах) и анализа тональности текста (анализ социальных медиа, идентификация позитивных и негативных мнений клиентов).

Рекомендательные системы. Наивный байесовский классификатор в сочетании с коллаборативной фильтрацией[2] (collaborative filtering) позволяет реализовать рекомендательную систему. В рамках такой системы с помощью методов машинного обучения и интеллектуального анализа данных новая для пользователя информация отфильтровывается на основании спрогнозированного мнения этого пользователя о ней.

2. Задача: Рубрикация новостных статей

Имеется коллекция новостных статей

$$D = \{d_1, d_2, \ldots, d_n\}$$

Имеется множество рубрик новостей

$$Y = \{y_1, y_2, \ldots, y_k\}$$

Необходимо построить модель, которая для заданной статьи определяет, к какой рубрики она относится

Векторное представление текстов

Словарь W – множество слов (после предобработки, нормализации, удаления стоп-слов), $|W| = m$

Документы D – множество статей, $|D| = n$

Статья x представляется как вектор

$$x = (x_1, x_2 \ldots, x_m)$$

Document-termmatrix$X$: строки – документы, столбцы – слова

[2]**Коллаборативная фильтрация** (англ. *collaborative filtering*)—это один из методов построения прогнозов (рекомендаций) в рекомендательных системах, использующий известные предпочтения (оценки) группы пользователей для прогнозирования неизвестных предпочтений другого пользователя.

Элементы матрицыx_{dt}

- Бинарные: $\begin{cases}1, t \in d\\0, t \notin d\end{cases}$
- Termfrequency (tf): сколько раз слово *t* встречается в документе*d* или производная от этого величина
- TF-IDF: $tfidf(t,d,D) = tf(t,d) \cdot idf(t,D)$,

$$idf(t,D) = log\frac{n}{1+|\{d \in D: t \in d\}|}$$

3. Байесовский классификатор

Задача: необходимо найти наиболее вероятное значение $y \in Y$ класса объекта $x = (x_1, x_2 \ldots, x_m)$ при условии заданных признаков объекта.

$$y(x) = \underset{y \in Y}{argmax} p(y|x)$$

Формула Байеса

апостериорная вероятность y при условии x $p(x|y)$ — *априорная вероятность класса y* $p(y)$

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)}$$

априорная вероятность объекта x $p(x)$

Максимум вероятности:

$$y(x) = \underset{y \in Y}{argmax} p(x|y)p(y)$$

Часто по умолчанию предполагают значения классов равновероятными

$$y(x) = \underset{y \in Y}{argmax} p(x|y)$$

Рубрикация новостных статей

$$y(x) = \underset{v \in Y}{argmax} p(x_1 = a_1, x_2 = a_2, \ldots, x_m = a_m|y = v)p(y = v)$$

Оценка $p(y = v)$: частота встречаемости статей рубрики *v* в коллекции $p(x_1 = a_1, x_2 = a_2, \ldots, x_m = a_m|y = v)$ - вероятность в точности такого набора слов. Оценить невозможно, т.к. нет такой статистики

Наивный Байес

Предположение об условной независимости атрибутов в общем случае неверно. Но эти зависимости совпадают для разных классов и «сокращаются» при оценке вероятностей. Например, грамматические зависимости остаются неизменными для всех рубрик новостей.

Наивный Байес хорошо показывает себя при решении задачи классификации, однако он не всегда может быть пригоден для оценки вероятностей.

Предположим условную независимость атрибутов при условии данного значения целевой функции

$$p(x_1 = a_1, x_2 = a_2, \ldots, x_m = a_m|y = v) =$$
$$= p(x_1 = a_1|y = v)p(x_2 = a_2|y = v) \ldots p(x_m = a_m|y = v)$$

Генеративная модель

Модель, по которой порождается документ:

MultinomialNaïveBayes: | Документ – это последовательность событий. Каждое событие –

BernoulliNaïveBayes:	это случайный выбор одного слова из словаря Документ – это вектор бинарных атрибутов, показывающих, встретилось ли в документе то или иное слово

Классификатор с мультиномиальным распределением (Multinomial Naïve Bayes)

Используется в случае дискретных признаков. Например, в задаче классификации текстов признаки могут показывать, сколько раз каждое слово встречается в данном тексте.

Пусть x_i - частота встречаемости слова *i* в документе

Тогда вероятность слова *i* в документе класса *v* оценивается как

$$\hat{\theta}_{vi} = \frac{N_{vi} + \alpha}{N_v + \alpha m}, где$$

$N_{vi} = \sum_{x:x\in D, y(x)=v} x_i \qquad N_v = \sum_{i=1}^{m} N_{vi} \alpha \geq 0$

Классификация:

$$y(x) = \underset{v\in Y}{argmax} \prod_{i=1}^{m} \hat{\theta}_{vi}^{xi}\, p(y = v)$$

Классификатор с распределением Бернулли (Bernoulli Naïve Bayes)

Используется в случае двоичных дискретных признаков (могут принимать только два значения: 0 и 1). Например, в задаче классификации текстов с применением подхода «мешок слов» (bagofwords) бинарный признак определяет присутствие (1) или отсутствие (0) данного слова в тексте.

Пусть $x_i \in \{0,1\}$ - встречается (1) или нет (0) слово *i* в документе

Тогда вероятность слова *i* в документе класса *v* оценивается как

$$P(x_i|y = v) = \frac{1 + N_{vi}}{2 + |\{x: x \in D, y(x) = v\}|}, где$$

$$N_{vi} = \sum_{x:x\in D, y(x)=v} x_i$$

Классификация:

$$y(x) = \underset{v\in Y}{argmax} p(y = v) \prod_{i=1}^{m} (p(x_i|y = v)a_i + (1 - p(x_i|y = v))(1 - a_i))$$

Особенности применения BernoulliNB и MultinomialNB

BernoulliNB принимает бинарные данные, MultinomialNB принимает счетные или дискретные данные (то есть каждый признак представляет собой подсчет целочисленных значений какой-то характеристики, например, речь может идти о частоте встречаемости слова в предложении). BernoulliNB и MultinomialNB в основном используются для классификации текстовых данных.

MultinomialNB и BernoulliNB имеют один параметр alpha, который контролирует сложность модели. Параметр alpha работает следующим образом: алгоритм добавляет к данным зависящее от alpha определенное количество искусственных наблюдений с положительными значениями для всех признаков. Это приводит к «сглаживанию» статистик. Большее значение alpha означает более высокую степень сглаживания, что приводит к построению менее сложных моделей. Алгоритм относительно устойчив к разным значениям alpha. Это означает, что значение alpha не оказывает значительного влияния на хорошую работу модели. Вместе с тем тонкая настройка этого параметра обычно немного увеличивает правильность.

4. Пример: 20 Newsgroups[3]

URL: http://qwone.com/~jason/20Newsgroups/

- Набор новостных статей «20 Newsgroups»
- 18000 новостных статей из 20 различных рубрик.

РешениевScikit-learn

```
from sklearn.datasetsimport fetch_20newsgroups
from pprintimport pprint
from sklearn.feature_extraction.textimport CountVectorizer
from sklearn.naive_bayesimport MultinomialNB
from sklearn.naive_bayesimport BernoulliNB
from sklearnimport metrics
from sklearn.metricsimport classification_report

# Загрузка данных для обучения и тестирования
newsgroups_train = fetch_20newsgroups(subset='train',
remove=('headers', 'footers', 'quotes'))
newsgroups_test = fetch_20newsgroups(subset='test')
news = fetch_20newsgroups(subset='all')

# Список новостных рубрик
# (совпадает в обучающей  и тестовой выборке)
print("Число наблюдений в обучающей выборке\n",
newsgroups_train.filenames.shape)
print("Число наблюдений в тестовой выборке\n",
newsgroups_test.filenames.shape)
print("Список новостных рубрик\n")
pprint(list(newsgroups_train.target_names))
pprint(list(newsgroups_test.target_names))
# Приведение данных к document-term матрице
vectorizer = CountVectorizer()
sparse_train = vectorizer.fit_transform(newsgroups_train.data)
sparse_test = vectorizer.transform(newsgroups_test.data)
# Размерность данных (в dense и sparse совпадает)
print("Размерность обучающей выборки sparse\n", sparse_train.shape)
print("Размерность тестовой выборки sparse\n", sparse_test.shape)
dense_train = sparse_train#.toarray()
dense_test = sparse_test#.toarray()
print("Размерностьобучающейвыборки dense\n", dense_train.shape)
print("Размерность тестовой выборки dense\n", dense_test.shape)
mnb = MultinomialNB(alpha= 1)
for iin range(0, 100):
# Используйте данные обучения для оценки параметров модели и
прогнозирования
mnb.fit(sparse_train, newsgroups_train.target)
pred = mnb.predict(sparse_test)
# Точностьклассификации
a = metrics.accuracy_score(newsgroups_test.target, pred, normalize=True)
print("Точность классификации – доля верно классифицированных объектов из
тестовой выборки \n", a)
print(classification_report(newsgroups_test.target, pred,
target_names=newsgroups_test.target_names))
clf = BernoulliNB(alpha=1)
for iin range(0, 100):
clf.fit(dense_test, newsgroups_test.target)
pred = clf.predict(dense_test)
```

[3] Данные «The 20 Newsgroups» — это коллекция примерно из 20000 новостных документов, разделенная (приблизительно) равномерно между 20 различными категориями. Коллекция «The 20 newsgroups» стала популярным набором данных для экспериментов с техниками машинного обучения для текстовых приложений, таких как классификация текста или его кластеризация.

```
# Точностьклассификации
a = metrics.accuracy_score(newsgroups_test.target, pred, normalize=True)
print("Точность классификации – доля верно классифицированных объектов из
тестовой выборки \n", a)
print(classification_report(newsgroups_test.target, pred,
target_names=newsgroups_test.target_names))
```

Результат работы программы:

```
Число наблюдений в обучающей выборке
 (11314,)
Число наблюдений в тестовой выборке
 (7532,)
Список новостных рубрик

['alt.atheism',
'comp.graphics',
 'comp.os.ms-windows.misc',
 'comp.sys.ibm.pc.hardware',
 'comp.sys.mac.hardware',
 'comp.windows.x',
 'misc.forsale',
 'rec.autos',
 'rec.motorcycles',
 'rec.sport.baseball',
 'rec.sport.hockey',
 'sci.crypt',
 'sci.electronics',
 'sci.med',
 'sci.space',
 'soc.religion.christian',
 'talk.politics.guns',
 'talk.politics.mideast',
 'talk.politics.misc',
 'talk.religion.misc']
['alt.atheism',
 'comp.graphics',
 'comp.os.ms-windows.misc',
 'comp.sys.ibm.pc.hardware',
 'comp.sys.mac.hardware',
 'comp.windows.x',
 'misc.forsale',
 'rec.autos',
 'rec.motorcycles',
 'rec.sport.baseball',
 'rec.sport.hockey',
 'sci.crypt',
 'sci.electronics',
 'sci.med',
 'sci.space',
 'soc.religion.christian',
 'talk.politics.guns',
 'talk.politics.mideast',
 'talk.politics.misc',
 'talk.religion.misc']
Размерность обучающей выборки sparse
 (11314, 101631)
Размерность тестовой выборки sparse
 (7532, 101631)
Размерность обучающей выборки dense
 (11314, 101631)
Размерность тестовой выборки dense
 (7532, 101631)
```

```
Точность классификации – доля верно классифицированных объектов из тестовой выборки
0.6355549654806161
                          precision    recall  f1-score   support

             alt.atheism       0.84      0.31      0.46       319
           comp.graphics       0.54      0.72      0.62       389
 comp.os.ms-windows.misc       0.20      0.00      0.01       394
comp.sys.ibm.pc.hardware       0.59      0.62      0.60       392
   comp.sys.mac.hardware       0.97      0.38      0.55       385
          comp.windows.x       0.39      0.89      0.54       395
            misc.forsale       0.93      0.59      0.72       390
               rec.autos       0.93      0.62      0.74       396
         rec.motorcycles       1.00      0.63      0.77       398
      rec.sport.baseball       1.00      0.63      0.78       397
        rec.sport.hockey       0.92      0.91      0.91       399
               sci.crypt       0.45      0.95      0.61       396
         sci.electronics       0.75      0.39      0.52       393
                 sci.med       0.64      0.84      0.73       396
               sci.space       0.73      0.86      0.79       394
  soc.religion.christian       0.49      0.93      0.64       398
      talk.politics.guns       0.69      0.66      0.68       364
   talk.politics.mideast       0.61      0.86      0.72       376
      talk.politics.misc       0.51      0.53      0.52       310
      talk.religion.misc       0.94      0.06      0.11       251

                accuracy                           0.64      7532
               macro avg       0.70      0.62      0.60      7532
            weighted avg       0.70      0.64      0.61      7532

Точность классификации – доля верно классифицированных объектов из тестовой выборки
0.780801911842804
                          precision    recall  f1-score   support

             alt.atheism       0.95      0.46      0.62       319
           comp.graphics       0.96      0.48      0.64       389
 comp.os.ms-windows.misc       0.89      0.85      0.87       394
comp.sys.ibm.pc.hardware       0.72      0.94      0.81       392
   comp.sys.mac.hardware       0.74      0.91      0.82       385
          comp.windows.x       0.94      0.86      0.90       395
            misc.forsale       0.37      0.98      0.54       390
               rec.autos       0.85      0.91      0.88       396
         rec.motorcycles       0.67      0.97      0.80       398
      rec.sport.baseball       0.87      0.89      0.88       397
        rec.sport.hockey       0.98      0.94      0.96       399
               sci.crypt       0.86      0.91      0.88       396
         sci.electronics       0.88      0.88      0.88       393
                 sci.med       0.90      0.74      0.82       396
               sci.space       0.94      0.87      0.90       394
  soc.religion.christian       0.73      0.82      0.77       398
      talk.politics.guns       0.80      0.76      0.78       364
   talk.politics.mideast       0.99      0.68      0.80       376
      talk.politics.misc       0.99      0.32      0.48       310
      talk.religion.misc       1.00      0.02      0.04       251

                accuracy                           0.78      7532
               macro avg       0.85      0.76      0.75      7532
            weighted avg       0.85      0.78      0.77      7532
```

5. Задания

Для набора данных «20 Newsgroups»

1. Подобрать оптимальное значение параметра α из интервала (0, 1)
2. Обучить классификатор с разными априорными вероятностями классов: равными и соответствующими долям классов в обучающей выборке

Работа № 3.5 Метод опорных векторов

1. Теоретические сведения

Метод опорных векторов (МОВ, англ. Support Vector Machine, SVM) - это техника машинного обучения с учителем. Она используется в классификации, может быть применена к регрессионным задачам.

Метод определяет границу принятия решения (далее ГПР) вместе с максимальным зазором, который разделяет почти все точки на два класса, оставляя место для неправильной классификации.

Цель МОВ — определить гиперплоскость (также называется «разделяющей» или ГПР), которая разделяет точки на два класса.

Для ее визуализации представим двумерный набор данных:

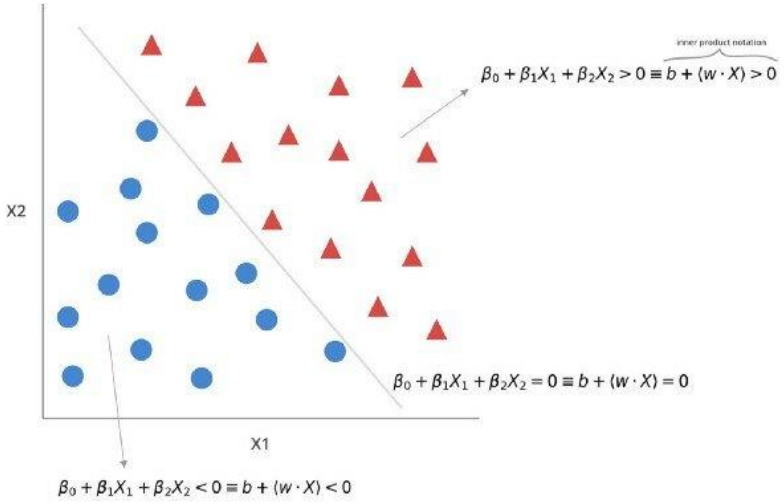


Рисунок. Гиперплоскость, которая полностью разделяет точки на двакласса.

Математическая постановка

Пусть даны два линейно разделимых класса объектов

Мы можем описать все точки разделяющей гиперплоскости используя вектор-нормаль к этой гиперплоскости:

$$\langle w, x \rangle - b = 0$$

Данная разделяющая гиперплоскость находится в «разделяющей полосе», которую мы также можем задать уравнениями:

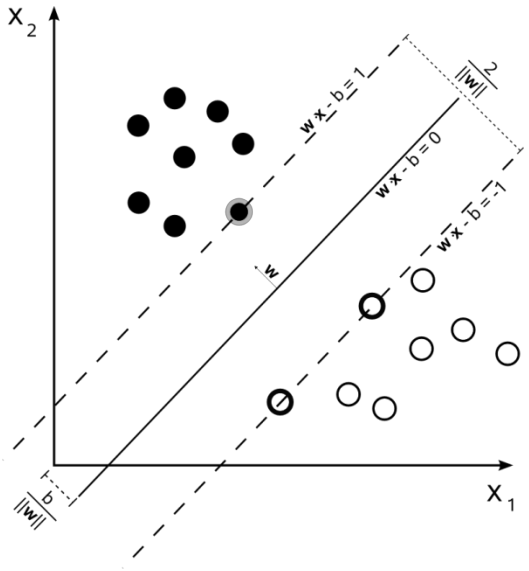
$$\langle w, x \rangle - b = -1 \qquad \langle w, x \rangle - b = 1$$

Можно найти ширину данной полосы как

Для машины опорных векторов необходимо найти разделяющую гиперплоскость, которая задает полосу максимальной ширины. Задача оптимизации:

$$\frac{1}{2}\|w\|^2 \to min$$

$$y_i(\langle w, x_i \rangle - b) \geq 1, \forall i = 1, \dots, n$$



Решение оптимизационной задачи

Для решения оптимизационной задачи с прошлого слайда можно воспользоваться известным методом множителей Лагранжа (Lagrangianrelaxation). Коэффициенты лямбда неотрицательны, поэтому если некоторые точки нарушают ограничения, заданные задачей, значение функции увеличивается. Устремив эту функцию к минимуму по w, b мы будем стремиться к тому, чтобы как можно больше точек не нарушало ограничения. Таким образом, 2-ая часть уравнения будет вносить неположительную часть в функцию и максимальное значение функции будет $\frac{\|w\|^2}{2}$. Максимизируя после этого по неотрицательным лямбда мы получим некоторую нижнюю границу оптимального значения.

$$L_P = \frac{\|w\|^2}{2} - \sum_{i=1}^{n} \lambda_i (y_i(< w, x_i > + b) - 1) \rightarrow \max_{\lambda_i \geq 0} \min_{w,b}$$

Взяв частные производные по w, b и приравняв к нулю, мы избавляемся от минимизации по этим переменным и можем заменить w = sum(y_i*lambda_i*x_i), 0 = sum(y_i*lambda_i). Можно перейти к двойственной проблеме, которая максимизируется по неотрицательным лямбда. Для этого нужно взять производные по переменным минимизации (w, b) и полученные значения подставить в прямую задачу. Получается двойственная задача, которую можно максимизировать только по лямбдам.

$$\frac{\partial L_P}{\partial w} = 0 \quad \rightarrow \quad w = \sum_{i=1}^{n} \lambda_i y_i x_i \, \frac{\partial L_P}{\partial b} = 0 \quad \rightarrow \sum_{i=1}^{n} \lambda_i y_i = 0$$

Для вычисления двойственной функции достаточно знать метки классов, а также скалярное произведение двух точек. Значения w и b можно затем найти после решения задачи для lambda.

Двойственная задача максимизации может быть решена с помощью градиентного спуска, где вектор лямбд итерационно обновляется с помощью вектора частных производных функции L_D.

$$L_D = \sum_{i=1}^{n} \lambda_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \lambda_j y_i y_j < x_i, x_j > \rightarrow \max_{\lambda_i \geq 0}$$

Случай линейно неразделимой выборки

Формулировка епсилон как функции потерь и ограничений снизу эквивалентна. Ошибка епсилон равна 0 если объект расположен за пределами “разделяющей” полосы.

Ошибка от 0 до 1 говорит о том, что объект расположен внутри полосы, но с правильной стороны от разделяющей гиперплоскости. Ошибка больше 1 говорит о том, что объект классифицирован неверно. Данная формулировка позволяет классифицировать объекты с ошибкой, что обязательно происходит для линейно неразделимой выборки.

Для неразделимых классов вводится функция потерь

$$\varepsilon_i = max\big(0, 1 - y_i(\langle w, x_i\rangle - b)\big)$$

Задача оптимизации:

$$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\varepsilon_i \to min$$

$$y_i(\langle w, x_i\rangle - b) \geq 1 - \varepsilon_i, \quad \varepsilon_i \geq 0, \qquad \forall i = 1, \dots, n$$

Данная форма SVM называется C-classification.

Метод множителей Лагранжа также решает оптимизационную задачу для случая C-classification. Добавляются новые слагаемые с переменной эпсилон и минимизация L_P происходит по трём переменным $-w, b, \varepsilon$. Решение оптимизационной задачи для постановки soft-margin мало отличается от исходного решения оптимизационной задачи, поскольку частные производные по w, b остаются те же самые. Добавляется лишь производная по эпсилон, которая даёт дополнительное ограничение на лямбда – лямбда от 0 до C. Поскольку бета также является множителем Лагранжа и больше или равно нуля, а лямбда также больше нуля, то лямбда не может быть больше C.

Подставив найденные производные в исходную задачу, мы получим ту же самую двойственную проблему, потому что все слагаемые с эпсилон сократятся. Максимизируя двойственную проблему по всем лямбда от 0 до C можно найти решение задачи.

- Метод множителей Лагранжа:

$$L_P = \frac{\|w\|^2}{2} + C\sum_{i=1}^{n}\varepsilon_i - \sum_{i=1}^{n}\lambda_i(y_i(< w, x_i > + b) - 1 + \varepsilon_i) - \sum_{i=1}^{n}\beta_i\varepsilon_i \to \max_{\lambda_i \geq 0}\min_{w,b,\varepsilon_i}$$

- Производные:

$$\frac{\partial L_P}{\partial w} = 0 \;\to\; w = \sum_{i=1}^{n}\lambda_i y_i x_i \qquad \frac{\partial L_P}{\partial b} = 0 \;\to \sum_{i=1}^{n}\lambda_i y_i = 0$$

$$\frac{\partial L_P}{\partial \varepsilon_i} = 0 \;\to\; C - \lambda_i = \beta_i \geq 0, \forall i$$

- Двойственная проблема:

$$L_D = \sum_{i=1}^{n}\lambda_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\lambda_i\lambda_j y_i y_j < x_i, x_j > \to \max_{0 \leq \lambda_i \leq C}$$

Предсказание

Функция предсказания представляет из себя просто определение расположения точки относительно разделяющей гиперплоскости. W заменена на значение своей производной. В функции предсказания также используется функция ядра (то есть простое скалярное произведение, либо одна из функций, его заменяющих) для внутреннего произведения двух точек. Если какая-то точка не лежит на границе «разделяющей» полосы, то максимальное значение функции будет достигаться если соответствующая лямбда = 0 (см. постановку прямой задачи с множителями лагранжа). Те точки, для которых лямбда не равно нулю и есть опорные, поскольку только от них зависит результат функции предсказания.

По сути, процесс обучения SVM на тренировочной выборке просто представляет из себя процесс решения двойственной задачи оптимизации.

Для предсказания результата алгоритма, используется функция sign:

$$F(z) = sign(\sum_{i=1}^{n} \lambda_i y_i \langle x_i, z \rangle + b)$$

Для $\lambda_i=0$ точка x_i не является «опорной», таким образом, в сумму, которая определяет класс нового объекта, влияние вносят только «опорные» точки.

Использование метода SVM

- Использован набор данных BanknoteAuthentification
- В качестве тестовой выборки взяты 107 последних объектов класса «0» (настоящие банкноты) и 119 объектов класса «1» (фальшивки). Остальные объекты используются в качестве обучающей выборки.
- Исходные параметры алгоритмов SVM взяты одинаковыми для разных библиотек.

Ядра (KernelTrick)

Вместо скалярного произведения точек в двойственной проблеме можно использовать специальные функции, называемые **ядрами**.

Линейное ядро—это аналог применения линейных преобразований к пространству объектов. Предположим, вы увеличиваете исходное пространство объектов возведением во вторую степень. Вы применили квадратичную функцию к исходному набору объектов. Теперь в этом расширенном пространстве есть оригинальная функция и ее квадратичная версия. Здесь неявно существует функция, которая сопоставляет эти два пространственных объекта.

$$x_1, x_2, x_3 \rightarrow x_1, x_1^2, x_2, x_2^2, x_3, x_3^2$$

Расширение пространства объектов с помощью квадратичной версии исходных.

Данные функции переводят пространство, в котором находятся наши точки в пространство большей размерности, что может привести к улучшенной разделимости классов. С полиномиальными ядрами вы проецируете исходное пространство объектов в полиномиальное. Граница, разделяющая классы, определяется полиномом более высокого порядка.

Использование ядер отличает классификаторы от метода опорных векторов, что открывает путь к решению более сложных задач. Но увеличение пространства признаков означает рост вычислительных требований. При большом пространстве функций подгонка модели станет дорогостоящей с точки зрения времени и ресурсов.

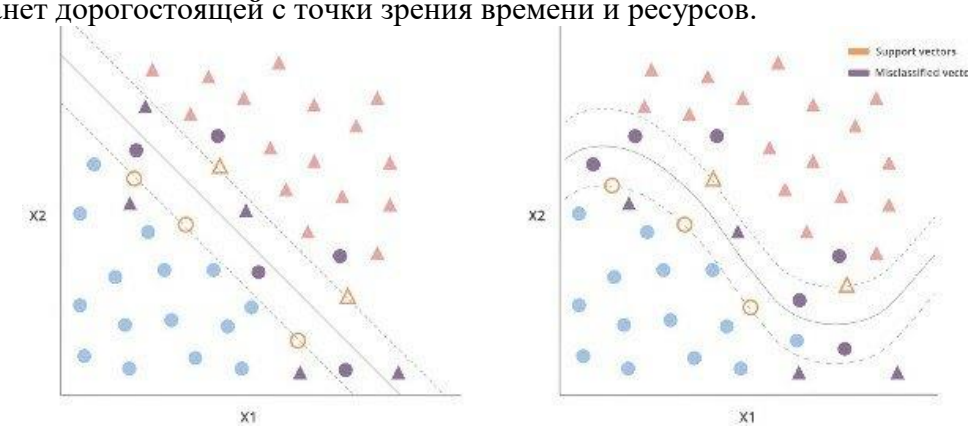


Рисунок. Граница решения и запасдля МОВ, наряду с соответствующими опорными векторами, используют линейное (справа) и полиномиальное ядро(слева).

Плюсы и минусы SVM

- Плюсы:
  - это наиболее быстрый метод нахождения решающих функций;

  - метод сводится к решению задачи квадратичного программирования в выпуклой области, которая всегда имеет единственное решение;
  - метод находит разделяющую полосу максимальной ширины (для заданных параметров), что позволяет в дальнейшем осуществлять более уверенную классификацию (и интерпретацию);
- Минусы
  - метод чувствителен к шумам и стандартизации данных;
  - не существует общего подхода к автоматическому выбору ядра, его параметров и построению спрямляющего подпространства в целом в случае линейной неразделимости классов.

2. Задача – выявление фальшивых банкнот

БазаBanknoteauthentification:
https://archive.ics.uci.edu/ml/datasets/banknote+authentication
Объекты представляют из себя характеристики изображений банкнот
- 1372 объекта
- 4 признака:
  - энтропия изображения
  - коэффициенты дисперсии, ассиметрии и эксцесса вейвлет-преобразования изображения
- Класс (фальшивые или настоящие)

Способ решения – классификация.
Методклассификации – SVM (Support Vector Machine).
Положительные стороны SVM:
- быстрый метод классификации;
- метод сводится к решению задачи квадратичного программирования в выпуклой области, которая обычно имеет единственное решение;
- метод позволяет осуществлять более уверенную классификацию, чем другие линейные методы.

Практическое задание

1. Проанализировать разные результаты для набора данных BanknoteAuthentification, в чём разница базовых настроек алгоритма в разных инструментах?
2. Найти наилучшие параметры для данных BanknoteAuthentification, используя технику кросс-валидации.
3. Возможно ли улучшить точность алгоритма для данных AdultIncome, используя другие параметры (gamma, C, параметры, связанные с SVM)? Подобрать параметры, дающие большую точность или показать, что для большого набора параметров точность улучшить не удаётся.

Работа № 3.6 Нейронные сети

Однослойная нейронная сеть

Нейронная сеть - это алгоритм обучения с учителем, который аппроксимирует функцию $f(\cdot): R^{in} \rightarrow R^{out}$ обучением на наборе данных, где in — количество измерений для ввода и out – размерность выхода.

Искусственный нейрон (ИН) – это простейший аналоговый преобразующий элемент, имитирующий поведение биологического нейрона (рисунок 1).

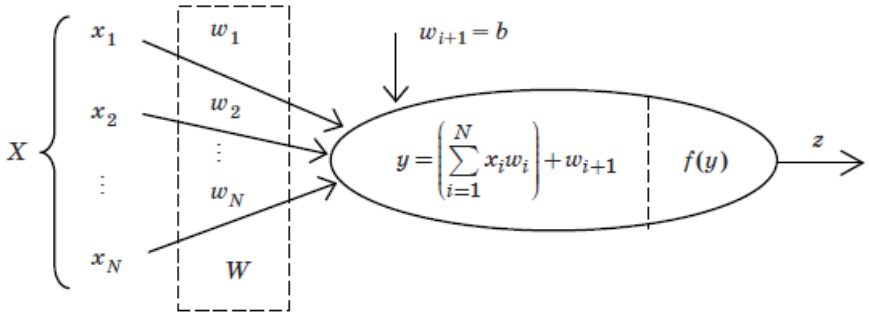


Рисунок 1. – Модель искусственного нейрона

На вход ИН поступает множество (вектор) сигналов. Каждый вход взвешивается – умножается на определенный коэффициент (весовой коэффициент). Сумма всех произведений определяет уровень активации нейрона. Суммирующий блок соответствует соме живого нейрона

Активационная функция F должна быть монотонной. Обычно $F(y)$ принадлежит к интервалу [0,1] или [–1,1]. Чаще используют сигмоидальная активационная функция (2).

Таким образом, ИН выполняет две операции. Сначала вычисляется сумма скалярного произведения вектора весов W и входного вектора X:

$$y = X^T W + b \tag{1}$$

Затем срабатывает активационная сигмоидная функция, определяющая значение выходного сигнала:

$$z = F(y) = \frac{1}{1+\exp(-k \cdot y)}. \tag{2}$$

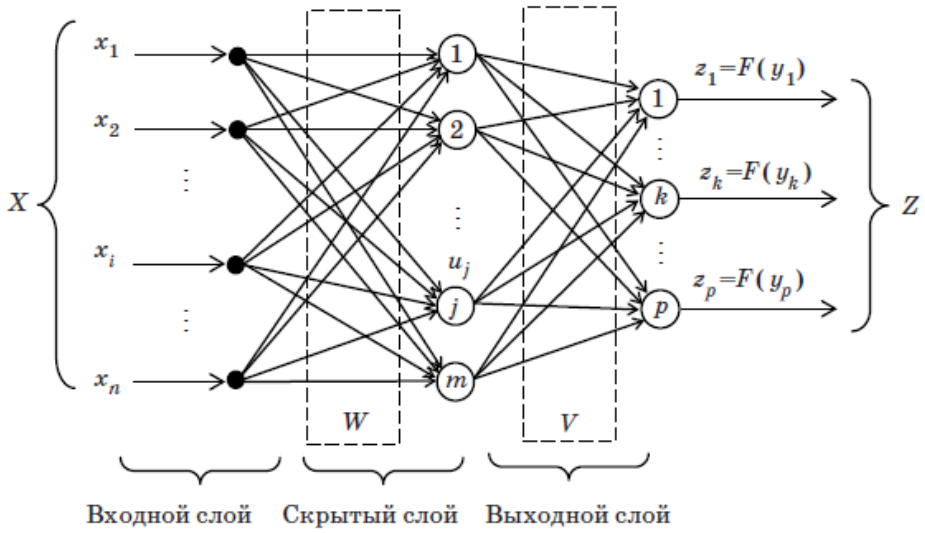


Рисунок 2. – Однослойная НС

Самый левый слой, известный как входной, состоит из набора нейронов $X = x_i|x_1, x_2, \dots, x_n$ представляющие входные функции. Каждый нейрон в скрытом слое преобразует значения из предыдущего слоя с взвешенным линейным суммированием $w_1x_1 + w_2x_2 + \cdots + w_mx_m$, за которой следует нелинейное функциональное преобразование - функция сигмоидного ограничения. Выходной слой получает значения из последнего скрытого слоя и преобразует их в выходные значения.

Нормирование означает приведение каждой компоненты входного вектора к интервалу [0, 1] или [–1, 1]. При известном диапазоне изменения входной переменной [x_{min}, x_{max}] нормирование выполняется по формуле:

$$x_i = \frac{x - x_{min}}{x_{max} - x_{min}} \quad (3)$$

Если значения входной переменной требуется привести к заданному интервалу [a, b], то применяется формула:

$$x_i = \frac{(x - x_{min})/(b-a)}{x_{max} - x_{min}} + a \quad (4)$$

При оценке качества работы ИНС обычно требуется рассматривать среднюю квадратическую ошибку (СКО), определяемую как усредненную на n примерах сумму квадратов разностей между желаемой величиной выхода z_i и реально полученными на сети значениями y_i для каждого i-го примера:

$$J = \frac{1}{N}\sum_{i=1}^{N}(z_m - y_i)^2 \quad (5)$$

Для решения задачи с применением НС, используйте реализации сетей, приложенные к заданию.

Метрики производительности

Для оценки производительности моделей прогнозирования используются следующие метрики:

- Errors = прогнозные значения – действительные значения
- MSE (Mean Squared Error) = $\frac{1}{n}\sum(\text{Прогноз} - \text{Действительныезначения})2$
- RMSE (Root Mean Squared Error) = $\sqrt{MSE}$
- MAE (Mean Absolute Error) = $\frac{1}{n}\sum|\text{Прогноз} - \text{Действительныезначения}|$
- MAPE (Mean Absolute Percentage Error) = $\frac{1}{n}\sum\frac{|\text{Прогноз}-\text{Действительныезначения}|}{\text{Действительныезначения}}$

В случаях, когда большие ошибки важны и они могут привести к серьезным последствиям, следует уделять больше внимания MSE. В том случае, когда приоритетом обладают малые ошибки, следует уделить внимание MAE.

4. Практическое задание

Для данных из работы 3.2 выполнить вычисления по модели нейронной сети прямого распространения и провести анализ полученной модели по СКО.

Качество оценки

	СКО
Линейная регрессия	
Нейронная сеть (конфигурация 1)	
…	
Нейронная сеть (конфигурация n)	

Контрольные вопросы

1. Что такое искусственная нейронная сеть?
2. В чем сходство и отличие линейной регрессии и НС прямого распространения?
3. Какие принципы используются при классификации нейронных сетей?
4. Каково определение искусственного нейрона?
5. Из каких составляющих состоит искусственный нейрон?
6. Какие варианты активационной функции могут быть использованы?
7. Каковы основные парадигмы обучения нейронных сетей?
8. Какие операции могут выполняться при предварительной обработке обучающих данных для нейросети?
9. Как оценить качество обучения нейросети?

Работа № 3.7. Бустинг (AdaBoost, LogitBoost, BrownBoost)

Теоретические сведения

Бустинг — это семейство ансамблевых алгоритмов, суть которых заключается в создании сильного классификатора на основе нескольких слабых. Для этого сначала создаётся одна модель, затем другая модель, которая пытается исправить ошибки в первой. Модели добавляются до тех пор, пока тренировочные данные не будут идеально предсказываться или пока не будет превышено максимальное количество моделей.

AdaBoost был первым действительно успешным алгоритмом бустинга, разработанным для бинарной классификации. Именно с него лучше всего начинать знакомство с бустингом. Современные методы вроде стохастического градиентного бустинга основываются на AdaBoost.

AdaBoost используют вместе с короткими деревьями решений. После создания первого дерева проверяется его эффективность на каждом тренировочном объекте, чтобы понять, сколько внимания должно уделить следующее дерево всем объектам. Тем данным, которые сложно предсказать, даётся больший вес, а тем, которые легко предсказать, — меньший. Модели создаются последовательно одна за другой, и каждая из них обновляет веса для следующего дерева. После построения всех деревьев делаются предсказания для новых данных, и эффективность каждого дерева определяется тем, насколько точным оно было на тренировочных данных.

Так как в этом алгоритме большое внимание уделяется исправлению ошибок моделей, важно, чтобы в данных отсутствовали аномалии.

1. Набор данных Bioresponse

https://www.kaggle.com/c/bioresponse

- 3751объектов, 1777признаков.
- Объект представляет из себя характеристики некоторой молекулы
- Класс объекта – вызвала ли молекула реакцию или нет
- Классификация. Способ классификации - Boosting

Предобработка данных

Пропуски в данных? Нет

Набор данных был разделен на обучающую и тестовую выборку.

Обучающая выборка содержит 3 000 объектов, а тестовая выборка – 751.

Проблема

Можно ли из полученных классификаторов построить агрегированный классификатор с предсказательной точностью выше, чем у найденных классификаторов?

Boosting

Бустинг — это семейство ансамблевых алгоритмов, суть которых заключается в создании сильного классификатора на основе нескольких слабых. Для этого сначала создаётся одна модель, затем другая модель, которая пытается исправить ошибки в первой. Модели добавляются до тех пор, пока тренировочные данные не будут идеально предсказываться или пока не будет превышено максимальное количество моделей.

AdaBoost

AdaBoost был первым действительно успешным алгоритмом бустинга, разработанным для бинарной классификации. Именно с него лучше всего начинать знакомство с бустингом.

Современные методы вроде стохастического градиентного бустинга основываются на AdaBoost.

AdaBoost используют вместе с короткими деревьями решений. После создания первого дерева проверяется его эффективность на каждом тренировочном объекте, чтобы понять, сколько внимания должно уделить следующее дерево всем объектам. Тем данным, которые сложно предсказать, даётся больший вес, а тем, которые легко предсказать, — меньший. Модели создаются последовательно одна за другой, и каждая из них обновляет веса для следующего дерева. После построения всех деревьев делаются предсказания для новых данных, и эффективность каждого дерева определяется тем, насколько точным оно было на тренировочных данных.

Так как в этом алгоритме большое внимание уделяется исправлению ошибок моделей, важно, чтобы в данных отсутствовали аномалии.

Алгоритм AdaBoost строит «сильный» классификатор вида:

$$F_T(x) = sign(f_T(x)) = sign(\sum_{t=1}^{T} \beta_t h(x, a_t))$$

где$w_t \in \mathbb{R}$, $h(x, a_t): X \times A \to \{-1,1\}$ - «слабый» классификатор, принадлежащий некоторому семейству классификаторов H, а A – пространство параметров этого семейства.

Пример H - одноуровневые деревья принятия решений.

Для нахождения классификатора $F_T(x)$ алгоритм AdaBoost последовательно ищет оптимальные параметры β_t и a_t $(1 \le t \le T)$ для построения классификатора $F_t(x)$, используя найденный ранее классификатор $F_{t-1}(x)$:

$F_t(x) = sign(f_{t-1}(x) + \beta_t h(x, a_t)), 1 \le t \le T$

при условии $f_0(x) = 0$.

Шаг 1: Инициализируем веса объектов:

$$w_i = \frac{1}{n}, i = 1, \dots, n$$

Шаг 2: Последовательное построение классификаторов $F_t(x), 1 \le t \le T$

Для *t = 1,...,T:*

a) Обучить слабый классификатор $h(x, a_t) \in H$, используя в качества функции потерь:

$$L(a) = \sum_{i=1}^{n} w_i \, I[\![h(x_i, a) \neq y_i]\!]$$

где $I[\![h(x_i, a) \neq y_i]\!]$ - индикатор ошибки.

b) Подсчет β_t:

$$\beta_t = \frac{1}{2} ln \frac{1 - L(a_t)}{L(a_t)}$$

c) Обновление весов объектов:

$$w_i = \frac{w_i e^{-\beta_t y_i h(x_i, a_t)}}{Z_t}, i = 1, \dots, n$$

где $Z_t -$ нормировочный множитель.

Шаг 3: Построение итогового классификатора:

$$F_T(x) = sign(\sum_{t=1}^{T} \beta_t h(x, a_t))$$

AdaBoost (Scikit-learn)

```
classifier =
ensemble.AdaBoostClassifier(DecisionTreeClassifier(max_depth=1),n_estimato
rs=10)
```

```
#обучениеклассификатора
classifier.fit(train[:,0:n],train[:,n])
#предсказание
prediction = classifier.predict(test[:,0:n])
tab = pd.crosstab(index = prediction, columns= test[:,n])

print(tab)
```

LogitBoost

- Основное отличие LogitBoost от AdaBoost состоит в том, что AdaBoost использует экспоненциальную функцию потерь, а LogitBoost – логистическую.
- За счет этого, в некоторых случаях LogitBoost может превосходить по точности AdaBoost, а также быть более устойчивым к шумам в данных.

$$y_i \in \{0,1\}$$

Шаг 1: Инициализируем переменные и вероятность того, что объект относится к классу 1:

$$f_T(x) = 0$$

$$w_i = \frac{1}{n} \text{ и } p(x_i) = \frac{1}{2}, i = 1, \dots, n$$

Шаг 2: Последовательное построение классификаторов $F_t(x), 1 \le t \le T$

Для *t = 1,...,T:*

a) Расчет w_i и z_i:

$$w_i = p(x_i)(1 - p(x_i))$$

$$z_i = \frac{y_i - p(x_i)}{p(x_i)(1 - p(x_i))}$$

b) Обучить $h(x, a_t) \in H$, используя в качества функции потерь:

$$L(a) = \sum_{i=1}^{n} w_i \, (h(x_i, a) - z_i)^2$$

c) Обновление $f_T(x)$ и $p(x)$:

$$f_T(x) = f_T(x) + \frac{1}{2} h(x, a_t) p(x) = (e^{f_T(x)})/(e^{f_T(x)} + e^{-f_T(x)})$$

Шаг 3: Построение итогового классификатора:

$$F_T(x) = \begin{cases} 1, \text{ если } sign(f_T(x)) \ge 0 \\ 0, \text{если } sign(f_T(x)) < 0 \end{cases}$$

BrownBoost

- В алгоритме BrownBoost используется дополнительная переменная – «время» работы алгоритма.
- Алгоритм BrownBoost более устойчив к шумам в данных (McDonald R. A. et al. An empirical comparison of three boosting algorithms on real data sets with artificial class noise).

$$y_i \in \{-1,1\}$$

Шаг 1: Подсчитываем «время» работы алгоритма c:

$$c = erfinv^2(1 - \varepsilon)$$

где ε – заданная точность классификации для функции потерь $\frac{1}{n}\sum_{i=1}^{n} |F_T(x_i) - y_i|$,

$erfinv$ – обратная функция к функции $erf(z) = \frac{2}{\pi}\int_0^z e^{-x^2} dx$

Шаг 2: Инициализируем «оставшиеся время» работы алгоритма s_1 и предсказанное значение $r_1(i)$ для объекта i:

$s_1 = c$ и $r_1(i) = 0, i = 1, \dots, n$

Шаг 3: Последовательное построение классификаторов $F_k(x)$

Для *k = 1,2,...* пока $s_k > 0$

a) Задание весов объектов:

$$w_i = \frac{e^{-(r_k(i)+s_k)^2/c}}{Z_k}, i = 1, \dots, n$$

где $Z_k -$ нормировочный множитель.

b) Нахождение слабого классификатора $h(x, a_k) \in H$ такого, что $\sum_{i=1}^{n} w_i h(x_i, a_k) y_i > 0$.

c) Для нахождения $t_k = t^* > 0$, $\beta_k = \beta^*$ таких, что $\gamma^* \leq \nu$ (ν – малая заданная константа, используемая для исключения вырожденных случаев) или $t^* = s_k$ решить дифференциальное уравнение с вещественными переменными γ, β, t:

$$\frac{dt}{d\beta} = \gamma = \frac{\sum_{i=1}^{n} \exp(-\frac{1}{c}(r_k(i) + \beta h(x_i, a_k) y_i + s_k - t)^2)\, h(x_i, a_k) y_i}{\sum_{i=1}^{n} \exp(-\frac{1}{c}(r_k(i) + \beta h(x_i, a_k) y_i + s_k - t)^2)}$$

с краевыми условиями $t = 0, \beta = 0$.

d) Задание s_{k+1} и $r_{k+1}(i)$:

$$r_{k+1}(i) = r_k(i) \beta_k h(x_i, a_k) y_i,\ i = 1, \dots, n$$

$$s_{k+1} = s_k - t_k$$

Шаг 4: Построение итогового классификатора:

$$F_T(x) = sign(\sum_{t=1}^{k-1} \beta_t h(x, a_t))$$

Практическое задание

1. Постройте график зависимости точности классификации и времени работы AdaBoost, LogitBoost, BrownBoost от максимального количества итераций.
2. Постройте график зависимости точности классификации и времени работы BrownBoost от значений параметров ε и ν.
3. Выполните предобработку данных:
   а) анализ и удаление выбросов
   б) снижение размерности
4. Оцените точность классификаторов, найденных AdaBoost, LogitBoost, BrownBoost.

Работа № 3.8. Кластеризация: алгоритмы K-means и EM

1. Теоретические сведения. Метод K-means

Был изобретён в 1950-х годах математиком Гуго Штейнгаузом и почти одновременно Стюартом Ллойдом. Особую популярность приобрёл после работы Маккуина.

- Изобретён в 1950-х годах
- Целевая функция - минимум суммы квадратов расстояний от точек до центров соответствующих им кластеров
- Смешанная (дискретно-непрерывная) задача оптимизации
- K-means – **эвристика**!
- Итерация алгоритма состоит из 2-х этапов
- Количество кластеров задаётся заранее

Описание алгоритма K-means

- **Инициализация**: алгоритм инициализируется центрами кластеров:
  - Первые k точек
  - Случайные k точек
  - Заранее определенные k точек
  - K-means++
  - Другиеалгоритмы (Random Partitioning, Build Algorithm…)
- **Шаг назначения**: известны центры кластеров. Распределение точек по ближайшим кластерам.
- **Шаг обновления**: пересчёт центров кластеров.

При инициализации точки как правило выбираются из набора данных. В некоторых случаях, точки могут быть заданы и не из набора данных, а как некоторые заранее определенные значения.

Если расстояние эвклидово – то среднее

Если Манхеттена – то медиана

K-means++ - каждый следующий центроид выбирается по вероятности – чем дальше расположена точка от текущих известных центров, тем выше вероятность ее выбора.

Назначение:

$$S_i^{t+1} = \{x_p : (x_p - \mu_i^t)^2 \leq (x_p - \mu_j^t)^2, \forall j \neq i\}$$

S_i– кластер с номером i, x– точки для кластеризации, μ– центры кластеров, t– номер шага.

Обновление средних:

$$\mu_i^{t+1} = \frac{1}{|S_i^{t+1}|} \sum_{x_j \in S_i^{t+1}} x_j$$

Оба шага таким образом уменьшают сумму квадратов расстояний.

Действительно, после шага 1 суммарное расстояние уменьшиться поскольку точки могли быть распределены в более дальние кластеры, таким образом их вклад может только уменьшиться. На втором шаге находится среднее значение всех точек, входящих в новый кластер, известно, что именно среднее значение даёт минимальную сумму квадратов расстояний от него до всех значений точек (минимум дисперсии достигается в мат. ожидании).

Алгоритм останавливается после определенного количества итераций либо по достижении сходимости (центры и распределение точек по кластерам не изменилось за итерацию)

K-means

Как правило нужно запускать алгоритм с разным k чтобы найти оптимальное разбиение

Например, вместо нахождения 3 кластеров (1 большой, 2 поменьше, но все явно отделены друг от друга), алгоритм может разбить большой кластер на 2 поменьше, а в третий поместить 2 маленьких.

Для улучшения результатов можно ограниченное число раз перезапустить алгоритм для других начальных центроидов

- Особенности метода:
  - Количество кластеров необходимо определять самостоятельно
  - Теоретически обеспечена сходимость к локальному минимуму
  - На практике локальные минимумы могут не давать логичный результат для кластеризации
  - Результат зависит от выбора начальных центроидов, которых может быть бесконечно много.
  - Старается создавать кластеры примерно одного размера / разброса, выделяет эллиптические(шарообразные) кластеры

1. Задача

- Набор данных –Quake (землетрясения)
- Задача – определить точки сейсмической активности
- Способ решения – кластеризация с выделением центров кластеров
- Методы кластеризации – K-means, EM-алгоритм

http://sci2s.ugr.es/keel/category.php?cat=uns

Набор данных Quakes (землетрясения)

- Объекты – информация о землетрясения
- 2178 объектов
- 4 признака:
  - Глубина точки гипоцентра землетрясения
  - Широта и долгота точки землетрясения
  - Сила землетрясения по шкале Рихтера
- Какую информацию можно найти в этой базе с помощью кластеризации?

```
from sklearnimport cluster
model = cluster.KMeans(n_clusters=n,init='random', algorithm='full',
max_iter=10000)
#Заданиепараметров
clusterobj = model.fit(dataset)
#Нахождениекластеров
print(clusterobj.cluster_centers_)
print(clusterobj.inertia_)
#Распечатка центров полученных кластеров и целевой функции
```

EM-алгоритм

Имя алгоритму было дано в 1977 году в статье авторов Arthur Dempster, Nan Laird, Donald Rubin. До этого метод использовался различными учёными для разных задач без какого-то определения алгоритма. Однако именно эта статья в *Journal of the Royal Statistical Society* дала жизнь EM-методу как инструменту статистического анализа.

- Данные представляются как выборка из смеси распределений (например, нормальных)
- Целевая функция – функция правдоподобия для предложенного набора данных и заданного количества компонент смеси.
- Итерация алгоритма состоит из 2-х этапов
- Количество компонент задаётся заранее

Описание EM-алгоритма

Далее предполагается что распределения нормальные (гауссовы). Под параметрами тогда понимаются вектор средних и матрица ковариаций каждой компоненты смеси, а также априорные вероятности того, из какой компоненты получены данные. В качестве скрытых переменных могут быть взяты переменные z_{ij}, которые представляют из себя апостериорные вероятности получения объекта I из компоненты j. При инициализации, их берут такими, что если объект i получен из компоненты j, то $z_{ij} = 1$, иначе = 0.

Тогда, на шаге E данные апостериорные вероятности могут быть получены через формулу Байеса –f это функции плотности, альфа – априорные вероятности.

На шаге M параметры пересчитываются с помощью весов W довольно очевидным образом.

- Предполагается, что данные X получены из смеси распределений с параметрами ϑ, также предполагается наличие скрытых переменных Z. Параметры перед началом алгоритма инициализируются.
- **Estimation (E-step):** На основе данных и известных параметров вычисляется матрица Z весов или апостериорных вероятностей:

$$w_{ij} = \frac{f\left(x_i \middle| z_{ij}, \theta_j^{s-1}\right) * \alpha_j^{s-1}}{\sum_{r=1}^{k} f(x_i | z_{ir}, \theta_r^{s-1}) * \alpha_r^{s-1}}$$

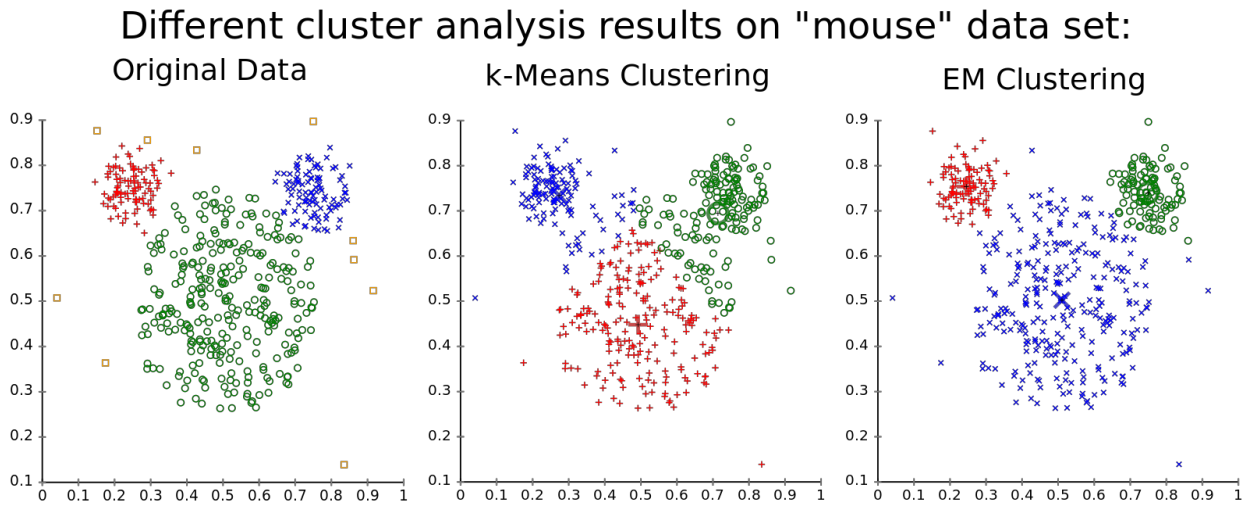
- **Maximization (M-step):** Пересчитываются параметры (средние значения, априорные вероятности, матрицы ковариации)

Особенности EM - алгоритма

Для большей устойчивости алгоритм можно несколько раз перезапускать с разными начальными условиями.

- Количество компонент необходимо определять самостоятельно
- Вектор скрытых переменных вводится таким образом, чтобы:
  - Его было легко найти при известных параметрах
  - Поиск максимума правдоподобия упрощается если известен вектор скрытых переменных
- Теоретически обеспечена сходимость к локальному минимуму
- Локальный минимум сильно зависит от начальной инициализации параметров (неустойчивость по начальным данным)

Картинка внизу хорошо подходит для K-means. Сверху можно увидеть пример не самого хорошего набора данных для K-means кластеризации – видно, что кластеры пытаются быть примерно одного размера



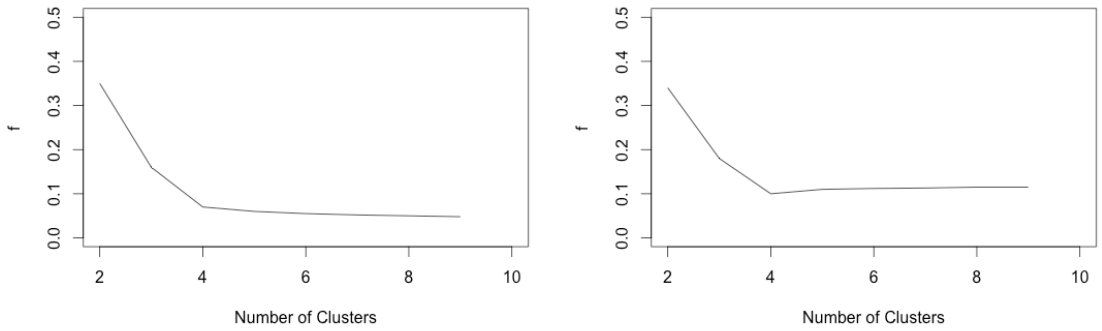
Алгоритмы определения количества кластеров

Вместо суммы внутрикластнерных расстояний можно использовать отношение среднего внутрикластерного расстояния к среднему межкластерному расстоянию.

Индексы как правило обладают некоторым набором значений, при котором можно сделать вывод о высокой степени кластеризации. Наличие соответствующих значений для некоторого выбранного количества кластеров может являться причиной оконгчательного выбора данного числа кластеров.

- Каменистая осыпь: анализируя график суммы внутрикластерных расстояний, эмпирически находится место, где увеличение количества кластеров перестаёт сильно влиять на изменение этой суммы.
- Различныеиндексы– Davies-Bouldin index, Dunn index, Silhouette coefficient.
- Другие эмпирические наблюдения (по количеству объектов в кластерах, по визуальным данным и т.д.).

Слева пример графика функции внутрикластерных расстояний, справа – отношения внутрикластерных расстояний к межкластерным. Можно заметить, что в районе 4 кластеров оба графика «останавливаются». Это может служить эмпирической причиной выбора 4 кластеров в этом конкретном случае




```
from sklearnimport mixture
model = mixture.GaussianMixture(n_components=50, max_iter=10000)
#Задание параметров
model.fit(dataset)
#Нахождение кластеров
print(model.means_)
#Распечатка центров полученных кластеров
```

Практическое задание

1) Построить графики показателей (сумма квадратов расстояний, отношение среднего внутрикластерного расстояния к внекластерному) для различного количества кластеров для набора данных Quake. Определить оптимальное число кластеров, анализируя эмпирическую информацию распределения «очагов».

Как правило нужно запускать алгоритм с разным k чтобы найти оптимальное разбиение

Например, вместо нахождения 3 кластеров (1 большой, 2 поменьше, но все явно отделены друг от друга), алгоритм может разбить большой кластер на 2 поменьше, а в третий поместить 2 маленьких.

Для улучшения результатов можно ограниченное число раз перезапустить алгоритм для других начальных центроидов