Лекция 1

Основные термины

kaggle.com - платформа для соревнований по анализу данных

Пример

Предсказание прибыли ресторана в зависимости от места размещения

Признаки:

Демография

1. Средний возраст жителей
2. Динамика количества жителей

….

x - объект, sample - то, для чего хотим сделать предсказание
X красивая - пространство всех возможных объектов
y - ответ, целевая переменная, target - что предсказываем
Y красивая - пространство ответов - все возможные значения ответов

Обучающая выборка
X некрасивое - обучающая выборка
l красивое - размер выборки

Объекты = абстрактная сущность, нужно задавать числовые характеристики = признаки
d - количество признаков
x = признаковое описание (вектор) = $(x_1, ..., x_d)$

Модель или алгоритм (hypothesis) - функция, предсказывающая ответ для любого объекта

Линейная модель: $A(x) = w_0 + w_1 x_1 + ... + w_d x_d$ где w - коэффициенты

Функция потерь → чтобы измерять насколько модель хорошая

Пример: $\text{квадратичное отклонение} = (a(x) - y)^2$, y - уже существующий факт

Функционал качества - мера качества работы алгоритма на выборке

Среднеквадратичная ошибка (Mean Squared Error, MSE)

$MSE = \frac{1}{\text{Количество объектов}} \sum (a(x) - y)^2$ → чем меньше, тем лучше

Функционал качества должен отвечать бизнес-требованиям

Обучение алгоритма
Есть обучающая выборка и функционал качества
Есть семейство алгоритмов
    Пример: все линейные модели
Обучение - выбор из всех моделей ту, которая выводит наименьшую среднюю ошибку

Машинное обучение не всегда сводится к этому
1. Обучение без учителя
2. Обучение с подкреплением

Зачем это нужно?

Сильный искусственный интеллект = может имитировать человека
Специализированный искусственный интеллект = решение отдельных и четких задач

Лекция 2

Обучение с учителем

Регрессия

Вещественные ответы: Y = R
Предсказание одного числа по другому числу

Классификация

Конечное число ответов → Ответы = классы, по которым хотим распределить объекты
Бинарная классификация: Y = {-1; +1} - ответы да или нет
Многоклассовая классификация → больше 2х классов
Пересекающаяся классификация (multi-label classification) → каждый набор может относиться к нескольким классам, можно закодировать через K нулей и единиц, обозначающих принадлежность к каждому классу

Ранжирование

Пример:
Набор документов d1, …, dn
Запрос q

Задача: отсортировать документы по релевантности запросу
a(q, d) - оценка релевантности

Обучение без учителя

Кластеризация

Y - отсутствует
Нужно найти группы похожих объектов

Типы признаков

D_j - множество значений признака

Бинарные признаки

$D_j = \{0, 1\}$

Вещественные признаки

$D_j = R \rightarrow$ числовые признаки

Категориальные (номинальные) признаки

D_j - неупорядоченное множество, значения которых нельзя сравнить между собой
Трудны в обращении, нельзя проводить простые арифметические операции

Гипотеза компактности и kNN - k nearest neighbors

Гипотеза компактности
Если два объекта близки друг к другу, то и ответы к ним близки друг к другу

kNN

Дано: обучающая выборка X
Задача классификации Y = {1, …, K}

Обучение модели = запоминаем обучающую выборку X

Применение

1. Считаем расстояние от нашего объекта до объектов в обучающей выборке
2. Сортируем по близости к новому объекту
3. Выбираем k ближайших объектов
4. Выбираем наиболее популярный среди них класс

$$a(x) = arg\,max \sum_{i=1}^{k} [y_{(i)} = y]$$

[..] - Аннотация Айверсона, логическое условие, которое выдает 1, если правда, 0 если ложь

Сравнение объектов и метрики

<u>Числовые данные</u>

Каждый объект описывается набором из d чисел - вектором

Если x - вектор, то xi - его i координата

Метрика - обобщение расстояния на многомерные пространства

Метрика - это функция с двумя аргументами, удовлетворяющая трем требованиям:

1. $p(x, z) = 0 \Leftrightarrow x = z$
2. $p(x, z) = p(z, x)$
3. $p(x, z) \leq p(x, v) + p(v, z)$

Евклидова метрика

$$p(x, z) = \sqrt{\sum_{j=1}^{d} (x_j - z_j)^2}$$

Манхеттенская метрика - не так сильно штрафует очень большие отклонения

$$p(x, z) = \sum_{j=1}^{d} |x_j - z_j|$$ - идем параллельно осям координат

Метрика Минковского

Через степени p

$$p(x, z) = \sqrt[p]{\sum_{j=1}^{d} (x_j - z_j)^p}$$

<u>Категориальные признаки</u>

Считающая метрика

Подсчет различий

$$p(x, z) = \sum_{j=1}^{d} [x_j \neq z_j]$$ → количество признаков по которым они отличаются

Счетчики

j-й признак: на какой категории чаще всего ездит пассажир

Посчитаем для каждой категории, как часто пассажиры соглашаются повысить класс
Заменяем категориальный признак на счетчики и считаем расстояния между ними

Функция потерь для классификации

Частный выбор - бинарная функция потерь
$L(y, a) = [a \neq y]$ - угадал/не угадал
Precision = точность
Accuracy = доля точных ответов
Баланс классов = отношение классов → разбалансированность может привести к низким значениям ошибки при плохой модели

У метода ближайшего соседа при k = 1 ошибка нулевая, но для новых объектов будут ошибки

Гиперпараметр

Нельзя подбирать k по обучающей выборке - *гиперпараметр*
Нужно использовать дополнительные данные

Лекция 3: Метод k ближайших соседей и линейная регрессия

Обобщающая способность

Как готовится к экзамену?

1. Заучить все примеры с занятий -- Переобучение -- Хорошее качество на обучающей выборке, низкое качество на новых данных
2. Разобраться в предмете и усвоить алгоритмы решения задач -- Обобщение -- Хорошее качество на обучающей выборке, хорошее качество на новых данных

Отложенная выборка

Часть выборки отводится для теста
Слишком большое обучение -- тестовая выборка нерепрезентативна
Слишком большой тест -- модель не сможет обучиться
Обычно: 70/30, 80/20

1. Надо перемешать выборку перед обучением
2. Мало объектов

Кросс-валидация

Разбиваем выборку на несколько кусков и потом мешаем их между тестовой и обучающей выборкой

→ можно найти гиперпараметры подбором того, при котором ошибка самая мальенкая

Leave-one-out -- Кросс-валидация, при которой число блоков = число объектов

Хороший, но медленный вариант

Если взять слишком большое k для ближайших соседей -- ответ будет один для всех -- модель становится константной -- количество ошибок увеличивается

Тестовая выборка -- мало обучающих элементов = много ошибок, среднее количество = оптимум, если число соседей большое = начинает расти вместе с ошибкой в обучающей выборке

После подбора всех гиперпараметров стоит проверить на новых данных, что модель работает

1. Обучающая выборка -- построение модели
2. Валидационная выборка -- подбор гиперпараметров модели
3. Тестовая выборка -- финальная оценка качества модели

Проблема с расстояниями -- может быть больше далеких соседей -- выдаст неверный ответ

Взвешенный knn

$$a(x) = arg\,max \sum_{i=1}^{k} w_i[y_{(i)} = y]$$

Метод Парзеновского окна

$$w_i = K(\frac{p(x, x_i)}{h})$$

K - ядро
h - ширина окна

Ядро - функция, которая задается на луче + чем больше аргумент, тем меньше ее значение
Обычно берется Гауссовское ядро -- стандартное нормальное распределение
Большой h - то далекие соседи тоже важны
Маленький h - важны только объекты рядом

kNN для регрессии

Задача регрессии - ответы из множества R

Обучение модели:

1. Запоминаем обучающую выборку

Применение модели:

1. Сортируем объекты по расстоянию
2. Берем среднее по соседям
3. Можно добавить веса, но надо нормировать по ним в конце, если сумма весов != 1

Функция потерь для регрессии

1. Частный выбор:
   a. $L(y,\ a)\ =\ (a\ -\ y)^2$
2. Функционал ошибки - среднеквадратичная ошибка
   a. $Q(a,\ X)\ =\ \frac{1}{l}\sum_{i\ =\ 1}^{d}(a(x_i)\ -\ y_i)^2$
3. Mean absolute error (MAE)
   a. $Q(a,\ x)\ =\frac{1}{l}\sum_{i\ =1}^{d}|a(x_i)\ -\ y_i|$ -- неудобно, так как не существует производной

Плюсы kNN

1. Очень простой
2. Легкое обучение
3. Мало гиперпараметров
4. Если данных много + хорошо подобрана функция расстояния → это лучшая модель
5. Бывает задача, где гипотеза компактности уместна
   a. Классификация изображений

Минусы kNN

1. Часто другие модели оказываются лучше
2. Надо хранить в памяти всю обучающую выборку
3. Искать k ближайших соседей довольно долго

Линейная регрессия

Парная регрессия

Простейший случай - один признак

Модель $a(x) = w_1 x - w_0$

Линейная - растет или убывает примерно со скоростью роста x

Два признака

$$a(x) = w_0 + w_1 x_1 + w_2 x_2$$

Ответ - плоскость

Общий случай

d признаков
d+1 параметр

$$a(x) = w_0\,(\text{сдвиг}) + w_1(\text{вес})\,x_1 + \ldots + w_d x_d$$

Лекция 4: Линейная регрессия

Скалярное произведение

Вектор a и вектор b

$$< a,\ b > = a_1 b_1 + a_2 b_2 + \ldots = ||a|| \times ||b|| \times cos(a,\ b)$$

$$a(x) = w_0 + w_1 x_1 + \ldots + w_d x_d = w_0 + < w,\ x >$$

Будем считать, что есть признак всегда равный 1, тогда w1 выполняет роль свободного коэффициента $\rightarrow w_1 \times 1 + w_2 x_2$= <w, x>

Применимость линейной регрессии

a(x) = <w, x>

1. Линейный рост прогноза
2. Необходимо, чтобы признаки были независимы
    a. a(x) = w_0 + w_1*(площадь) + w_2 * (район) + w_3 * (расстояние до метро)
        i. Растет линейно с ростом площади
        ii. Район -- категориальный признак
        iii. Расстояние до метро может быть нелинейным

Кодирование категориальных признаков

1. Значения признака район U = {u1, …, um}
2. Новые признаки вместо x_j: $[x_j = u_1]$, ..., $[x_j = u_m]$-- бинарные признаки, что район соответствует некоторому одному
3. **one-hot, dummy кодирование**

Кодирование нелинейной функции

Вводятся индикаторные признаки с попаданием x в промежуток

$w_3 \times [t_0 < x < t_1] + \ldots + w_{3+n}[t_{n-1} < x < t_n]$→ w3 -- вес = стоимость от расстояния

a(x) = w_0+ w_1 * (площадь) + w_2 * (квартира в ЦАО?) + w_3 * (квартира в ЮАО?) … → веса = прибавка от района

Линейная регрессия в векторном виде

Среднеквадратичная ошибка и задача обучения

$$\frac{1}{L}\sum_{i=1}^{l}(< w, x_j > - y_i)^2 \rightarrow min_w$$

Матрица - таблица с числами

Матрица “объекты признаки” -- столбец = признак, строка = элемент данных
Вектор-строка, вектор-столбец, …
Можно записать вектор весов

К применить модель к обучающей выборке?

Перемножить матрицу “объекты-признаки” на вектор весов

Вычисление ошибки

Евклидова норма

$$||z|| = \sqrt{\sum_{j=1}^{n} z_i^2}$$

np.square(X.dot(w) - y).mean()

Обучение линейной регрессии

Среднеквадратичная ошибка

$$Q(w_1, \ldots, w_d) = \text{MSE}$$

Производная
Показывает скорость роста функции
В точке минимума или максимума = 0

Градиент показывает в какую сторону от точки функция растет быстрее всего

Можно посчитать градиент MSE

$$= \frac{2}{L} \times X^T(Xw - y)$$

Приравниваем нулю и решаем СЛУ:

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

Слишком ресурсозатратный способ решения
Найти обратную матрицу = d^3

100 признаков = 1 секунда
1000 признаков = 20 минут
1000000 признаков = 31000 лет

Аналитическая формула -- слишком долго

Переобучение и регуляризация линейных моделей

Симптом переобучения

1. Большие коэффициенты -- симптом переобучения -- можно запретить модели иметь большие веса

Регуляризация

Штрафуем за большие веса

Регуляризатор:

$||w||$ = сумма весов^2 (L2 регуляризатор)

$$\frac{1}{l}\sum_{i=1}^{l}(< w, x_i > - y_i)^2 + \lambda \sum_{i=1}^{d} ||w||^2 \rightarrow min_w$$

Либо сумма модулей весов (L1 регуляризатор) → LASSO

$$\frac{1}{l}\sum_{i=1}^{l}(< w, x_i > - y_i)^2 + \lambda \sum_{i=1}^{d} |w_j|$$

Поощряет зануление весов → отбрасывает недостаточно важные коэффициенты → приводит к отбору признаков

Регуляризованный функционал:

$$MSE - \lambda \times ||w||^2 \rightarrow min_w$$

Лямбда - коэффициент регуляризации -- гиперпараметр

Аналитическое решение:

$$w = (X \times X^T - \lambda \times w)$$

→ **Гребневая регрессия (Ridge regression)**

Предсказание стоимости квартиры

Чем больше вес != тем важнее признак, так как у признаков могут быть разные порядки величин

→ Надо масштабировать все признаки → **нормализация**

$$x = \frac{x - \mu}{\sigma}$$

Правильнее всего оценивать важность признака, выкидывая по очереди все признаки и посмотреть как без них изменяется ошибка

Лекция 5: линейная регрессия и градиентный спуск

Градиент и его свойства

Градиент -- вектор частных производных

У градиента есть важное свойство →

1. Зафиксируем точку x_0
2. Градиент показывает в какую сторону происходит наискорейшее возрастание
3. Минус градиент -- наискорейшее убывание
4. В математике нет методов, которые позволяют найти глобальный минимум

Градиентный спуск
Сдвигаемся в точку, где функция убывает быстрее всего, потом повторяем алгоритм до тех пор, пока не оказываемся в точке, которая является локальным минимумом (некуда сдвигаться)

Парная регрессия
Функционал ошибки = … → трехмерная функция
Вектор градиента = набор из частных производных

Начальное приближение

w^0 -- инициализация весов
Можно выбрать прямую, которая связывает две точки
Или = корреляцию

Градиентный спуск

Повторять до сходимости:

$$w_t = w_{(t-1)} - \eta \times \nabla(w_{(t-1)})$$

Если градиент = 0, можно остановится, так как это будет точка минимума или максимума

Если $w_t - w_{(t-1)} < \epsilon$

η -- позволяет контролировать скорость обучения

Если сделать длину шага недостаточно маленькой, градиентный спуск может перешагнуть минимум

Переменная длина шага → подбор в зависимости от t

f'(x) < 0 ⇒ убывает
f'(x) > 0 ⇒ возрастает

Проблемы градиентного спуска

1. Находит только локальные минимумы →
    a. Можно несколько раз проводить процедуру из разных точек
    b. Метод отжига → берем в окрестности точки случайную точку

Масштабирование признаков → позволяет убрать колебания

Стохастический градиентный спуск

Оценка градиента
Считаем не сумму по объектам, а выбираем один объект

1. Начальное приближение w_0
2. Повторять, каждый раз выбирая случайный объект it
3. Останавливаемся, если
$$||w_t - w_{(t-1)}|| < \epsilon$$

Нет никаких гарантий, что найдется минимум + больше шагов + чувствительность к выбросам → надо модифицировать

Этот метод можно починить, используя переменную длину шага → примерно линейная скорость убывания → Надо подобрать гиперпараметр эта от т → быстрее перебирать

Mini-batch

Надо брать m случайных объектов

Функция потерь в градиентном спуске

На выбросе MSE сильно поднимается → среднее MSE ухудшается → модель склоняется к выбросу, за счет ухудшения точности на всей остальной выборки

Чтобы избежать такого → берется средняя абсолютная ошибка (MAE) → mean absolute error → модуль вместо квадрата

У модуля нет производной → не получится делать градиентный спуск

Функция потерь Хубера → где ошибки маленькие = считаются как квадрат, если отклонение большое, то считается как модуль

Лекция 6: Линейная классификация

Модель линейной классификации

$Y = \{-1, +1\}$

-1 -- отрицательный класс

+1 -- положительный класс

$a(x)$ должен выводить одно из этих чисел

Линейный классификатор $a(x) = sign(w_0 + \sum_{j=1}^{d} w_j x_j)$

w_0 - свободный коэффициент, w_j - веса, x_j - признаки

Будем считать, что есть единичный признак

$$a(x) = sign(\sum_{j=1}^{d} w_j x_j) = sign(< w, x >)$$

<w, x> < 0 → объекты слева от разделяющей гиперплоскости

<w, x> > 0 → объекты справа от разделяющей гиперплоскости

Расстояние от точки до гиперплоскости $\frac{|<w, x>|}{||w||}$

Отступ

$M_i = y_i < w, x_i >$

$M_i < 0$-- классификатор дает неверный ответ

$M_i > 0$-- классификатор дает верный ответ

Чем больше отступ, тем больше уверенность алгоритма в ответе

Порог классификатора

$a(x) = sign(< w, x_i > - t)$

Обучение линейных классификаторов

Функция потерь в классификации

Бинарная функция потерь

$$L(y, a) = [a \neq y]$$

Функционал ошибки -- доля ошибок (error rate)

$$Q(a,\ x)\ =\ \frac{1}{l}\sum_{i=1}^{l}[a(x_i)\ \neq y_i]$$

Доля верных ответов (accuracy):

$$Q(a,\ x)\ =\ \frac{1}{l}\sum_{i=1}^{l}[a(x_i)\ =\ y_i]$$

Доля ошибок для линейного классификатора

$$Q(a,\ x)\ =\ \frac{1}{l}\sum_{i=1}^{l}[sign(<w,\ x_i>)\ \neq y_i]$$

Индикатор -- недифференцируемая функция, что усложняет процесс обучения.
Решение: Оценить сверху дифференцируемой функцией и минимизировать функцию оценки, надеемся, что она прижмет долю ошибок к нулю

$$0\ \leq\ \frac{1}{l}\sum_{i=1}^{l}[y_i<w,\ x_i>\ \ <\ 0]\ \leq\ \frac{1}{l}\sum_{i=1}^{l}\hat{L}(y_i<w,\ x_i>)\ \rightarrow min_w$$

Примеры функций оценки

1. Логистическая $\rightarrow \hat{L}\ =\ log(1\ +\ e^{-M})$
2. Кусочно-линейная $\rightarrow \hat{L}\ =\ max(0,\ 1\ -\ M)$
3. Экспоненциальная $\rightarrow \hat{L}\ =\ e^{-M}$
4. Сигмоидная $\rightarrow \hat{L}\ =\ \frac{2}{1+e^{M}}$

Алгоритм обучения - тот же самый градиентный спуск по функции оценки. Важно **не добавлять** w_0 в регуляризацию. Можно использовать L1-регуляризацию.

Метрики качества классификации

1. Доля неправильных ответов $\frac{1}{l}\sum_{i=1}^{l}[a(x_i)\ \neq y]$
2. Доля правильных ответов (**accuracy**) $\frac{1}{l}\sum_{i=1}^{l}[a(x_i)\ =\ y_i]$

   Несбалансированных выборки
   1. Объектов одного из классов значительно больше
   2. Доля правильных ответов будет высокой, даже если модель будет все объекты относить к одному (большому) классу
   3. $accuracy\ \in [q_0;\ 1]$, где q_0 - доля большого класса

   Такие метрики не учитывают цену ошибок

Улучшение метрики:

Доля правильных ответов: r_1 и r_2

Абсолютное улучшение: $r_2 - r_1$

Относительное улучшение: $\frac{r_2 - r_1}{r_1}$

Precision

$$precision(a, X) = \frac{TP}{TP + FP}$$

Полнота (recall)

$$recall(a, X) = \frac{TP}{TP + FN}$$

Объединить точность и полноту

Максимизировать обе метрики одновременно удобно

1. *Арифметическое среднее* $A = \frac{1}{2}(precision + recall)$ → *не очень точно*
2. Минимум $M = min(precision, recall)$
3. F-мера $F = (1 + \beta^2)\frac{precision \times recall}{\beta^2 \times precision + recall}$
   a. $\beta < 1$ → важнее точность
   b. $\beta > 1$ → важнее полнота
4. Геометрическое среднее $G = \sqrt{precision \times recall}$

Лекция 7: Линейная классификация 2

Линейный классификатор

$a(x) = sign(< w, x > - t) = 2[< w, x > > t] - 1$

<w,x> - оценка принадлежности классу

Оценка принадлежности:

Пример: кредитный скоринг

b(x) -- оценка вероятности возврата кредита

a(x) = [b(x) > 0.5]

precision = 0.1, recall = 0.7

Если бы алгоритм оценивал корректно, то вероятность была бы близка к 0.5

PR-кривая

Кривая точности-полноты

Если модель никого не относит к положительному классу → в мире считается что точность = 1 → кривая стартует из 1

AUC -- area under curve → показатель качества модели

ROC-кривая

Ось X -- False positive rate → доля ложных срабатываний

$$FPR = \frac{FP}{FP + TN}$$

Количество ложных срабатываний / все отрицательные объекты

Ось Y -- TPR → доля верных срабатываний

$$TPR = \frac{TP}{FN + TP}$$

Если никого не отнесли к позитивному классу, то FRP = TPR = 0

Левая точка (0,0)
Правая точка (1,1)
Идеальная модель соответствует точке (0,1)
AUC-ROC → площадь под ROC-кривой → взаимно однозначно относятся

Идеальный алгоритм AUC-ROC = 1
AUC-ROC = 0, если перевернуть прогнозы модели, то она станет идеальной
ROC → чувствительно к размеру положительного класса и балансировке выборки

Логистическая регрессия

1. Решаем задачу бинарной классификации $Y = \{-1; 1\}$
2. Минимизация верхней оценки
   a. $\frac{1}{l}\sum_{i=1}^{l} log(1 + exp(-y_i < w, x_i >)) \rightarrow min_w$

Модель b(x) предсказывает вероятности, если среди объектов с b(x) = p доля положительных равна p

Линейный классификатор
Переводим выход модели на отрезок [0, 1]

С помощью сигмоиды → $\frac{1}{1 + exp(<w, x>)}$

Как обучать?

1. Если y_i = + 1, $\sigma(< w, x_i >) \to 1$ или <w, x_i > → + ∞
2. Если = -1 → устремляем к 0 → <w, x_i > → - ∞

Функционал ошибки

$$-\sum_{i=1}^{l}\{[y_i = 1]\sigma(< w, x_i >) + [y_i = -1](1 - \sigma(< w, x_i >)\} \to min_w$$

Если $y_i = +1$ и $\sigma(< w, x_i >) = 0$, то штраф = 1

Недостаточно строго

К сигмоидам добавляются *логарифмы*

$$-\sum_{i=1}^{l}\{[y_i = 1]log(\sigma(< w, x_i >)) + [y_i = -1]log((1 - \sigma(< w, x_i >))\}$$

- log-loss функция

Такая функция очень сильно штрафует, если модель уверена в неправильном ответе

$$\sum_{i=1}^{l} log(1 + exp(- y_i < w, x_i >)$$

Лекция 8: SVM

Hinge loss

1. Решаем задачу бинарной классификации $Y = \{+1, -1\}$
2. Минимизируем верхнюю оценку $\frac{1}{l} max(0, 1 - y_i < w, x >) \to min_w$
3. Отступ классификатора -- расстояние от ближайшего объекта до гиперплоскости
4. Максимизируем отступ классификатора + минимизируем ошибку

Простой случай → ***линейно-разделимая выборка***

1. Существует линейный классификатор, не допускающий ни одной ошибки

Требования:

1. $y_i(< w, x_i > + w_0) > 0 \ \forall i = 1, ..., l$ → не допускает ошибок
2. Максимальный отступ классификатора

i. Расстояние от точки до плоскости $\frac{|<w, x_i> + w_0|}{||w||}$, w не включает w_0

ii. Отступ классификатора: $min_{i = 1, ..., l} \frac{|<w, x_i> + w_0|}{||w||}$

iii. Линейный классификатор: $a(x) = sign(< w, x_i > + w_0)$ при делении на константу > 0 не изменяется

iv. Поделим w и w_0 на $min_{i = 1, ..., l} | < w, x_i > + w_0|$

v. Тогда: $min_{i = 1, ..., l} | < w, x_i > + w_0| = 1$

vi. Тогда отступ классификатора:

$$= \frac{min_{i = 1, ..., l}|<w, x_i> + w_0|}{||w||} = \frac{1}{||w||}$$

b. Тогда требование: $\frac{1}{||w||} \to max_w$

i. При условии: $min_{i = 1, ..., l} | < w, x_i > + w_0| = 1$

$$\{ ||w||^2 \to min_{w, w_0}$$

$$\{ y_i (< w, x_i > + w_0) \geq 1$$

Линейно-неразделимый случай

Любой классификатор допускает хотя бы одну ошибку

$$\{||w||^2 \to min_{w, w_0}$$

$$\{y_i (< w, x_i > + w_0) \geq 1 + \xi$$

$$\{\xi \geq 0$$

Но модель надо штрафовать за ошибки → вводим элемент штрафа в минимизацию:

Если C большое число → важна точность, маленькое - важен отступ, а не точность

$$\{||w||^2 + C \sum_{i = 1}^{l} \xi_i \to min_{w, w_0, \xi_i}$$

$$\{yi (< w, x_i > + w_0) \geq 1 + \xi$$

$$\{\xi \geq 0$$

Объединим ограничения:

$$\{||w||^2 + C \sum_{i = 1}^{l} \xi_i \to min_{w, w_0, \xi_i}$$

$$\{\xi_i \geq max(0,\ 1\ -\ y_i(< w,\ x_i >\ \ +\ w_0))$$

Тогда метод опорных векторов:

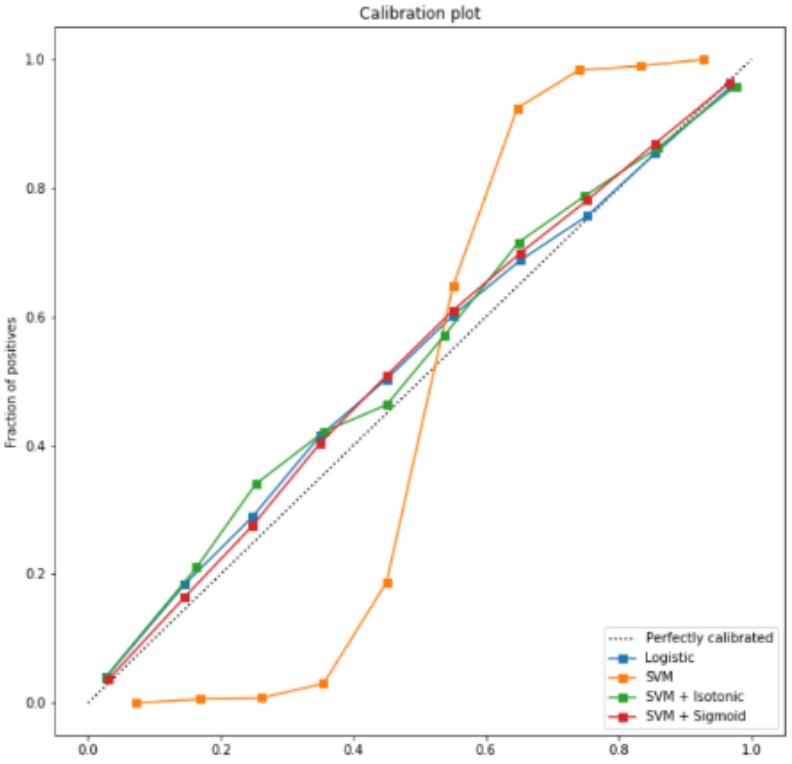
$$C\sum_{i=1}^{l} max\ (0,\ 1\ -\ y_i(< w,\ x_i >\ \ +\ w_0))\ +\ ||w||^2 \rightarrow min_{w,\ w_o}$$

Калибровка вероятностей

Модель b(x) предсказывает вероятности тогда, когда среди объектов с ответом b(x) = p доля положительных p

Калибровочная кривая

1. Разбиваем отрезок на промежутки
2. Для каждого промежутка берем объекты, для которых $b(x)\ \in [t_i;\ t_{i+1}]$
3. Считаем среди них долю положительных объектов



Для SVM большие отклонения по вероятности, поэтому его надо откалибровать.

Калибровка модели

Надо найти такое преобразование c(b(x)), которое выпрямит вывод модели по вероятности

Калибровка Платта

Обучаем логистическую регрессию на выборке $c(b(x_i),\ y_i)_{i=1}^{l}$

Единственный признак

$$c(b(x_i),\ y_i) = \frac{1}{1+ exp(p \times b(x) + q)}$$

$$-\sum_{i=1}^{l}([y_i = +1]log(c(b(x_i)) + [y_i = -1]log(1 - c(b(x_i))) \rightarrow min_{p,\,q}$$

Не стоит строить калибровку на тех же данных, на которых обучалась модель
На обучающей выборке хорошо приблизит, но новых данных b(x) будет иметь другое распределение
Используется кросс-валидация
b(x) строится на обучающем множестве, параметры c(x) подбираются на тестовом
Получаем столько моделей, сколько блоков в кросс-валидации → можно усреднить

Изотоническая регрессия

Обучающая выборка такая же как в Платте
Подбираем такую функцию, которая наиболее близка к наблюдениям и возрастает. То есть $b(x_1) > b(x_2) \rightarrow c(b(x_1)) > c(b(x_2))$

Тоже необходимо использовать кросс-валидацию или отложенную выборку

Многоклассовая классификация

One-vs-all	**All-vs-all**
Отделяем каждый класс от всех остальных	Отделяем каждый класс от каждого
K классов $Y = \{1;\ ...;\ K\}$	←
$X_k = (x_i;\ [y_i = k])_{i=1}^{l}$	$X_{km} = \{(x_i,\ y_i) \in X \mid y_i = k \text{ или } y_i = m\}$
Обучаем $a_k(x)$ на $X_k;\ k = 1,...,\ K$ (k классификаторов)	Обучаем k^2классификаторов
$a_k(x)$должен выдавать оценки принадлежности классу	←
$a_k(x) = arg\,max_{k=1,\,...,\,K}\,a_K(x)\rightarrow$ выбираем тот классификатор, который наиболее уверен в своем ответе	$a(x) = arg\,max_{k\in\{1,\,...,\,K\}}\sum_{m=1}^{K}[a_{km}(x) = k]$ → Относим к тому классу за который чаще голосовали классификаторы

Доля ошибок

Функционал ошибки - доля ошибок

$$Q(a, x) = \frac{1}{l} \sum_{i=1}^{l} [a(x_i) \neq y_i]$$

Можно измерить долю верных ответов

Микро-усреднение	Макро-усреднение
Вычисляем TP, FP, FN, TN для каждого класса	Вычисляем нужную метрику для всех классов
Суммируем их	Подставляем в формулу $precision/recall$
Подставляем в формулу $precision/recall$	Усредняем по всем классам
Крупные классы вносят больший вклад	Игнорирует размер классов

Решающие деревья

Линейная модель не подходит + может быть сложно проанализировать результаты и они могут быть не независимы → никогда не знаем какой из полиномиальных признаков важен = квадраты и парные произведения.
При полиномиальных признаках очень быстро растет количество признаков
Добавляем признаки, которые позволяют задать принадлежность интервалу → таких признаков становится еще больше

Решающие деревья

Непонятно по какому признаку определять интервалы
Логические правила - нужно определить в модели
Каждая вершина либо конечная, либо из нее торчит два ребра

Вершины из которых торчат ребра = внутренние вершины

То что в них записано = предикат
Листовые вершины = прогнозы
Деревом можно идеально решить любую задачу, но если оно переобученное, то на тестовой выборке будет плохо

Лекция 10 Решающие деревья 2

Различать хорошие вершины и плохие можно с помощью *энтропии*

$$H(p_1, \ldots, p_K) = -\Sigma p_i log_2 p_i$$ - доля хаотичности вершины

Критерий информативности

Какой предикат лучше -- сравниваем хаотичность в дочерних вершинах и в исходной

$$Q(R, j, t) = H(R_M) - (\frac{|R_l|}{|R|} H(R_l) + \frac{|R_r|}{|R|} H(R_r)) \to min_{j,t}$$

Умножаем энтропию на долю объектов в вершине
Для регрессии можно считать дисперсию

$$H(R) = \frac{1}{|R|}\Sigma(y_i - y_R)^2 \to$$ R - количество объектов в исходной вершине

Жадное построение дерева

- Оптимальный вариант -- перебрать все возможные решающие деревья, выбрать самое маленькое (мин количество вершин) среди безошибочных
- *Очень долго* → np-полная задача. На данный момент для них неизвестны решения кроме перебора, но не доказано что нельзя решить более эффективно. Требуют полиномиальное время для решения

  Как строить дерево?
  1. Умеем выбирать лучший предикат для разбиения вершины
  2. Будем строить жадно
  3. Начинаем с корня, разбиваем последовательно, пока не выполняется *критерий останова*
     a. Ограничить глубину → самый длинный путь от вершины до листа
     b. Ограничить количество листьев
     c. Задать минимальное число объектов в вершине
     d. Задать минимальное уменьшение хаотичности при разбиении

  Алгоритм

1. Строим корневую вершину и помещаем в нее всю выборку $R_1 = X$
2. Запустить построение из корня SplitNode(1, R_1)
    a. SplitNode(m, R_m)
        i. Если выполнен критерий останова, то выход
        ii. Ищем лучший предикат $j, t = arg\,min_{j,t} Q(R_m, j, t)$ -- суммарная энтропия в дочерних вершинах, которые получаются если вершину m разбить по признаку j и порогу t
            1. В качестве порогов используем все имеющиеся в разбиваемой вершине объекты
            2. Предикат может иметь вид через скалярное произведение, но это не имеет смысла
        iii. Разбиваем с помощью предиката объекты: $R_l = \{\{(x, y) \in R_m | [x_j < t]\}$, $R_r = \{\{(x, y) \in R_m | [x_j \geq t]\}$
        iv. Повторяем для дочерних вершин -- SplitNode(l, R_l), SplitNode(r, R_r)

Строим дерево

1. Проецируем выборку на оси
2. Разбиваем по первой оси -- мин энтропия 0.53
3. Разбиваем по второй оси -- мин энтропия 0.47
4. Лучший вариант на второй оси → разбиваем по ней

Жадный -- на каждом шаге берем лучший вариант и не меняем дерево

Вывод

1. Позволяют строить сложные деревья
2. Алгоритм сложный и требует перебора всех вариантов на каждом шаге

Работа с категориальными признаками

One-hot encoding (dummy encoding)

$$x_j: [x_j = u_1], \ldots, [x_j = u_m]$$

Нужно создавать много признаков

Mean target encoding

$$x_j = U_j = \{u_1, \ldots, u_m\}$$

Считаем сколько раз встречается в обучающей выборке:

$$count(j, u_p) = \sum_{i=1}^{l} [x_{ij} = u_p]$$

Для *регрессии* считаем суммарный ответ в категории:

$$target(j,\ u_p)\ =\ \sum_{i=1}^{l}[x_{ij}\ =\ u_p]\,y_i$$

Заменяем категориальный признак на числовой:

$$\widehat{x}_{ij} = \frac{target(j, x_{ij})}{count(j, x_{ij})}$$

Для *классификации* считаем классы в категории:

$$target_k(j,\ u_p)\ =\ \sum_{i=1}^{l}[x_{ij}\ =\ u_p][y_i = k]$$

Заменяем категориальный признак на K числовых:

$$\widehat{x}_{ij} = \left(\frac{target_1(j, x_{ij})}{count(j, x_{ij})},\ \ldots,\ \frac{target_K(j, x_{ij})}{count(j, x_{ij})}\right)$$

Если признак один для какого-то примера → признак является просто ответом в данном случае → **утечка целевой переменной** → модель может отбросить все остальные признаки

1 вариант → Надо испортить признак, чтобы ответ не так сильно коррелировал → подбираем шум, как гиперпараметр

2 вариант → добавление априорных величин в счетчики $\hat{x}_{ij} = \frac{target + a}{count + b}$

3 вариант → кросс-валидация счетчиков → вычислять ответ и count и target на разных блоках

Лекция 11 Композиция моделей

Неустойчивость деревьев

Устойчивость - если данные изменились не сильно, то и модель не должна сильно измениться
Обучаем дерево несколько раз на 90% из исходных данных
Получается N деревьев с разными ответами

Для классификации

N базовых моделей
Каждая модель лучше случайного угадывания
Объединяем их через голосование большинством:

$$a(x) = arg\,max_{y \in Y} \sum_{n=1}^{N} [b_n(x) = y]$$

Таким образом, получается более устойчивая модель
Если голоса поровну, то либо выдать наугад, либо выдать отказ от классификации, либо использовать нечетное количество моделей

Для регрессии

N базовых моделей
Усредняем наблюдения
Каждая модель хотя бы немного лучше случайного угадывания

Усредняем прогнозы: $a_N(x) = \frac{1}{N} \sum_{n=1}^{N} b_n(x)$

Как строить базовые модели:

1. Обучать независимо на разных подвыборках → ***Бэггинг*** → bootstrap aggregating
2. Обучать последовательно, чтобы каждая следующая исправляла ошибки предыдущих → *бустинг*

Бэггинг

Бутстрап

1. Выборка с возвращением
2. Берём i элементов из X
3. Какие-то объекты войдут несколько раз, а какие-то ни разу
4. В подвыборке будет i объектов и 63.2% уникальных
5. *Подмножество объектов*

Случайные подпространства

1. Выбираем случайное подмножество признаков
2. Обучаем модель только на них
3. Может быть плохо, так как могут быть очень важные признаки, без которых невозможно построить разумную модель
4. *Подмножество признаков*

Разложение ошибки на смещение и разброс

Ошибка модели раскладывается на три компоненты:

1. Шум - характеристика сложности и противоречивости данных → теоретическое свойство данных
2. Смещение (bias) - способность модели приблизить лучшую среди всех возможных моделей → насколько модель сложная по сравнению с задачей

3. Разброс (variance) → устойчивость модели к изменения в обучающей выборке

Бэггинг

Смещение $a_N(x)$, такое же, как у $b_n(x)$

Разброс $a_N(x)$:

$$\frac{1}{N}(\text{разброс}\, b_n(x)) \; + \; \text{ковариация}(b_n(x),\, b_m(x))$$ - две случайные модели

Если базовые модели независимы, то разброс уменьшается в N раз

Если корреляция есть, то разброс будет уменьшаться слабее от бэггинга

Получение низкой ошибки в бэггинге:

1. Взять сильно несмещенные модели (глубокие деревья)
2. Как можно менее связанные модели

Случайный лес

Жадный алгоритм

1. На втором шаге ищем лучший предикат среди случайной части признаков
2. Чем меньше признаков, тем меньше корреляции
3. Рекомендации для q:
   a. Регрессия: $q = \frac{d}{3}$
   b. Классификация $q = \sqrt{d}$

Для n = 1, …, N

1. Сгенерировать выборку $\hat{X}$ с помощью бутстрапа
2. Построить решающее дерево $b_n(x)$ по выборке
3. Дерево строится, пока в каждом листе не окажется не более n_{min}объектов
4. Оптимальное разбиение ищется среди q случайных признаков, *которые выбираются заново при каждом разбиении.*
5. Регрессия -- усреднение
6. Классификация -- голосование по большинству
7. Гиперпараметры Random Forest -- q -- число деревьев в итерации, n_{min}-- критерий останова, N -- число деревьев. Но их особо не нужно подбирать.
8. При росте числа деревьев Random Forest не переобучается → всегда используем много
9. Очень медленный метод

Суррогатные предикаты - в каждой вершине ищем два предиката, лучший и которой дает примерно такое же разбиение по другому признаку. При пропуске в данных используем запасной признак

Out-of-bag

Каждое дерево обучается примерно на 63% признаков
Остальные объекты - как бы тестовая выборка для дерева
X_n – обучающая выборка для $b_n(x)$
Можем оценить ошибку на новых данных

$$Q_{test} = \frac{1}{l}\sum_{i=1}^{l} L(y_i, \frac{1}{\sum_{n=1}^{N}[x_i \notin X_n]}\sum_{n=1}^{N}[x_i \notin X_n]b_n(x)) \text{ - средний}$$

ответ по i объектам, деревьев которые по ним не обучались

Важность признаков

Берем j признак и перемешиваем его значения. Смотрим насколько ошибка растет на испорченной выборке
Если качество сильно упало → признак очень важен для модели

Градиентный бустинг

Проблемы бэггинга

1. Если базовая модель окажется смещенной, то и композиция не справится с задачей
2. Базовые модели долго обучать и дорого хранить

Идея бустинга

1. Возьмем простые базовые модели
2. Будем строить композицию последовательно и жадно
3. Каждая модель строится так, чтобы максимально корректировать ошибки предыдущих моделей

$$a_N(x) = \sum_{n=1}^{N} b_n(x)$$

Обучение первой модели

$$\frac{1}{l}\sum_{i=1}^{l} L(y_i, b_1(x_i)) \rightarrow min_{b_1(x)}$$

Обучение N-й модели

$$\frac{1}{l}\sum_{i=1}^{l} L(y_i,\ a_{N-1}(x_i)\ +\ b_N(x_i))\ \to min_{b_N(x)}$$

$$a_{N-1}(x_i)\ \to \text{ответы предыдущих моделей},\ const$$

Нужно упростить задачу, так как непонятно как обучать дерево для такого случая

Случай MSE:

$$\frac{1}{l}\sum_{i=1}^{l}(a_{N-1}(x_i)\ + b_N(x_i)\ -\ y_i)^2\ \to min_{b_N(x)}$$

Сгруппируем:

$$\frac{1}{l}\sum_{i=1}^{l}(b_N(x_i)\ -\ (y_i\ -\ a_{N-1}(x_i)))^2\ \to min_{b_N(x)}$$

$$b_N(x_i)\ -\ \text{модель},\ y_i\ -\ a_{N-1}(x_i)\ =\ s^{(N)}{}_i\ (\text{сдвиги})\ -\ \text{константа}$$

$$\frac{1}{l}\sum_{i=1}^{l}(b_N(x_i)\ -\ s^{(N)}{}_i)^2\ \to min_{b_N(x)}$$

В градиентном бустинге рост количества деревьев приводит к росту ошибки, начиная с какого-то момента → надо подбирать нужное количество по кросс-валидации или на отложенной выборке

Проблемы для произвольной функции потерь

Обучаться на остатки?

$$\frac{1}{l}\sum_{i=1}^{l} L(y_i\ -\ a_{N-1}(x_i),\ b_N(x_i))\ \to min_{b_N(x)}$$

Логистическая функция потерь

$$a_N(x)\ =\ sign\sum_{n=1}^{N} b_N(x)$$

$$L(y,\ z)\ =\ log(1\ +\ exp(-\ yz))$$

$$\frac{1}{l}\sum_{i=1}^{l} log(1\ +\ exp(-\ (y_i\ -\ a_{N-1}(x_i))b_N(x_i))\ \to min_{b_N(x)}$$

Если ответ идеальный, то *объект не участвует в обучении*. N модель может выдать любой ответ

Иначе $s_i^{(N)} = \pm 2$ → поделим на 2 → модель учится выдавать корректный класс → требуем чтобы на всех ошибочных объектах выдавался максимальный отступ → требуем слишком многого!

Лекция 13: Понижение размерности данных

Проблема без понижения размерности:

Биоинформатика → 20000 признаков
Категориальные признаки
Анализ текстов
Анализ ЭКГ

Хотим -- сгенерировать новую матрицу с меньшим числом признаков

Зачем:

1. Проклятие размерности → сложно найти закономерности, объекты менее похожи друг на друга
2. Плохие признаки →
    a. коррелирующие признаки (лишняя информация)
    b. шумовые признаки -- выборка по ним неразделима, принимают случайное значение. Никак не связаны с целевой переменной. Случайный признак может хорошо коррелировать с целевой переменной, но на тестовых данных покажет плохое качество
3. Ускорение модели → менее сложные модели, могут быть ограничения на скорость модели

Методы понижения размерности:

1. Методы отбора признаков (feature selection) → убирают ненужные
    a. Фильтрация - без учета модели
    b. Обертка - надстройки на модели
    c. Понижение с помощью моделей
2. Извлечение признаков (feature extraction) → объединяют старые признаки в новые

Одномерные методы

1. Относятся к методам фильтрации и отбора признаков

x_{ij} – значение j – го признака на i – ом объекте

$\bar{x_j}$ – среднее значение j – го признака

y_i – значение целевой переменной на i – м объекте

$\bar{y}-$ среднее значение целевой переменной

Считаем насколько признак хороший подсчетом выборочной дисперсии.

$R_j = \frac{1}{l}\sum_{i=1}^{l}(x_{ij} - \bar{x})^2 \rightarrow$ признаки с низкой дисперсией не влияют на целевую переменную.

Можно считать корреляцию - $R_j = \frac{\sum_{i=1}^{l}(x_{ij} - \bar{x_j})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{l}(x_{ij} - \bar{x_j})^2 \sum_{i=1}^{l}(y_i - \bar{y})^2}}$

T-score

$$R_j = \frac{|\mu_1 - \mu_2|}{\sqrt{\frac{\sigma^2_1}{n_1} + \frac{\sigma^2_2}{n_2}}}$$

Для задач бинарной классификации

Проблемы подхода:

1. Может быть синергичность признаков

Лекция 14: Градиентный бустинг

Для произвольной функции потерь

$a_N(x) = \sum_{n=1}^{N} b_N(x)$

Обучение N-той модели

$\frac{1}{l}\sum_{i=1}^{l} L(y_i, a_{N-1}(x_i) + b_n(x_i)) \rightarrow min_{b_N(x)}$

Для логистической функции потерь:

$L(y, z) = log(1 + exp(-yz))$

Если

$y_i = a_{N-1}(x) \rightarrow$ объект в принципе не участвует в обучение и выдает любые ответы

Если модель ошибается, то разница будет ± 2

Делим на 2, чтобы решить проблему с ± 2

$\frac{1}{l}\sum_{i=1}^{l} log(1 + exp(-\frac{y_i - a_{N-1}(x_i)}{2} b_N(x)) \rightarrow min_{b_N(x)}$

При больших ошибках логистическая функция будет ошибаться, так как выдает просто правильный ответ

MSLE

$L(y, z) = (log(z + 1) - log(y + 1))^2$, работает для положительных чисел

$$\frac{1}{l}\sum_{i=1}^{l}(log(b_N(x_i) + 1) - log(y_i - a_{N-1}(x_i) + 1))^2 \to min_{b_N(x)}$$

Градиентный бустинг в общем виде

Куда и как сильно сдвигать ответ модели, чтобы уменьшить ошибку?

Посчитать производную этой функции

$$s_i^{(N)} = -\frac{\delta}{\delta z}L(y_i, z)\Big|_{z = a_{N-1}(x_i)}$$ → шаг по градиенту

Обучение N-модели

$$\frac{1}{l}\sum_{i=1}^{l}((b_N(x_i) - s_i^{(N)})^2 \to min_{b_N(x)}$$ ⟶ тут уже MSE и все хорошо максимизируется

Бустинг уменьшает смещение базовых моделей, однако разброс может увеличиться. Стоит брать неглубокие деревья.

Гиперпараметры

1. Глубина деревьев
2. Число деревьев

Проблемы бустинга

Сдвиги могут плохо обучиться на базовую модель → предлагается каждую модель с небольшим коэффициентом, чтобы она не сильно портила модель.

Рандомизация

Можно обучать на случайных подмножествах признаков или объектов → убираем шумовые объекты

Лекция 15: Отбор признаков

Отбор с помощью моделей

Линейные модели

Веса можно использовать для оценки важности признаков
Для повышения числа нулевых весов можно использовать L1-регуляризацию

Решающие деревья

Смотрим важность признаков = суммируем уменьшения энтропии по всем вершинам, где разбиение делалось про признаку j

Случайный лес → сумма по всем деревьям

Метод главных компонент

1. Метод извлечения признаков
2. Principal component analysis

Проецируем данные на признак с большой дисперсией

Линейный подход:

$$z_{ij} = \sum_{k=1}^{D} w_{jk} x_{ik}$$

t-SNE

t-Stochastic Neighbor Embedding

Ищет такие точки на плоскости, которые сохраняют расстояния изначального графика

Лекция 16: Кластеризация

Обучение без учителя

Данные не размечены → модель сама ищет в них закономерность

Кластеризация

Дана матрица объекты признаки X

Каждый кластер состоит из похожих объектов
Объекты из разных кластеров существенно отличаются
Найти:

Множество кластеров Y
Алгоритм кластеризации приписывает каждый объект к одному из кластеров

1. Нет строгой постановки
2. Множество кластеров неизвестно
3. Правильные ответы отсутствуют

Виды кластеризации

…
Иерархическая кластеризация
Жесткая и мягкая - один или несколько кластеров для каждого объекта

Чтобы проверить результаты нужно провести разметку

K-Means

Дано: выборка $x_1, \ldots, x_l$

Параметр - число кластеров
Начало: случайно выбрать K центров кластеров $c_1, \ldots, c_K$

Повторять по очереди до сходимости

1. Относим каждый объект к ближайшему центру
2. Переместить центр каждого кластера в центр тяжести

a. $$c_j = \frac{\sum_{i=1}^{l} x_i [y_i = j]}{\sum_{i=1}^{l} [y_j = j]}$$

Сходимость → если в результате шагов кластеры не поменялись и центры не поменялись

Выбор числа кластеров

Качество кластеризации - внутрикластерное расстояние

$$J(C) = \sum_{i=1}^{l} p(x_i, c_{y_i})$$

Зависит от K

Нужно подобрать такое K, после которого качество меняется не слишком сильно

Метод локтя

Ищем где происходит перелом графика

Особенности K-Means

Распараллеливается
Подходит для кластеров с простой геометрией
Требует выбора числа кластеров

Плотностные методы кластеризации

DBSCAN

Вводятся типы точек:

1. Шумовые - в окрестности нет основных
2. Основные - в окрестности ϵ n точек
3. Граничные - в окрестности меньше n точек, но есть основные

Вводятся переменные:

ϵ — окрестность, n — количество точек в окрестности

Алгоритм

1. Объекты ни к чем не отнесены
2. Выбираем точку без метки
3. Если в ее окрестности меньше N точек, то помечаем точку как шумовую и начинаем заново
4. Если это не так - создаем новый кластер
    a. Рассматриваем все точки из окрестности данного кластера
    b. Если новые точки граничные, то ничего не делаем
    c. Если новые точки шумовые, то относим его к шумовому кластеру
    d. А если новые точки основные, то распространяем их кластер на соседние
5. Переходим к шагу 1

Размер окрестности это гиперпараметр

Особенности DBSCAN

Находит кластеры произвольной формы
Можно работать с большими объемами данных
Нужно подбирать ϵ и n

Графовые методы

Граф - вершины соответствуют объектам
Соединяем ребрами близкие объекты
Выделяем компоненты связности

Текстовая кластеризация

Word2Vec → Слова со схожим смыслом часто идут в паре с одними и теми же словами

Контекст - окрестность слова

Хотим представить каждое слово в виде вектора

Требования:

Размерность должна быть не очень большой

Похожие слова должны иметь близкие векторы

Арифметические операции имеют смысл

Если два слова часто идут рядом, то их векторы сонаправлены

Если редко - ортогональны

Лекция 17: Рекомендательные системы

История:

1. 90-е → GroupLens
2. 2000-е → коммерциализация, закрытые исследования
3. 2006 → Netflix Prize - большой датасет для соревнования
    a. Первое соревнование
    b. Использование метрики RMSE
4. 2007 → RecSys - конференция

На основе чего строить рекомендации:

1. Данные по другим пользователям
2. Данные по объектам

Базовая логика рекомендательной системы:

1. Объект: пара “user-item”
2. Целевая переменная - клики, длинные клики, досмотры, покупки, ...
3. Особенности
    a. Выбор целевой переменной
    b. Метрики качества необычные
    c. Факторы для модели
    d. Очень большие объемы данных

Memory-based models

Обозначения

1. Множество товаров: I
2. Множество пользователей: U

3. Множество пар “пользователь-товар”, для которых известны оценки: *R*

Оценки

1. Явные → лайки, дизлайки
2. Неявные → долго смотрел
3. Неявные оценки более шумные

Сходство пользователей:

$I_{uv} = \{i \in I \mid \exists r_{ui}$ и $\exists r_{vi}\}$- множество товаров, которые оценили и пользователь u и пользователь v

Тогда сходство пользователей:

$$w_{uv} = \frac{\sum(r_{ui} - \bar{r}_u)(r_{vi} - \bar{r}_v)}{\sqrt{\sum(r_{ui} - \bar{r}_u)^2}\sqrt{\sum(r_{vi} - \bar{r}_v)^2}}$$

$\bar{r}_u, \bar{r}_v$ – средние рейтинги пользователей

Средние позволяют центрировать рейтинги

User-based collaborative filtering

1. Приходит пользователь u_0
2. Находим пользователей, которые похожи на него
   a. $U(u_0) = \{v \in U \mid w_{u_0 v} > \alpha\}$
3. Рекомендуем те товары, которые часто выбирали пользователи из $U(u_0)$

Недостатки

1. Непонятно какой взять порог
2. Нужно строить матрицу айтемов - оценок = минус память
3. Непонятно какую метрику близости использовать

Модели со скрытыми переменными

Векторы интересов

Для пользователя - Оцениваем насколько нравится каждая из категорий

Для фильма - оцениваем принадлежность к категориям в виде вектора

$\text{Заинтересованность} = \text{пользователь} \times \text{фильм}$ -- скалярное произведение

Как это обучать?

Обучаем на известных оценках векторы так, чтобы скалярное произведение хорошо предсказывало оценки.

Как правило эти модели объединяют и строят еще что-нибудь более сложное сверху

Лекция 18: Ранжирование

Формализация

1. Дан набор запросов $\{q_1, \ldots, q_m\}$
2. Дан набор документов $\{d_1, \ldots, d_n\}$
3. Нужно для каждого запроса правильно упорядочить документы
4. Рассматриваем пары запрос-документ (q, d)
5. Для некоторых троек (q, d_1, d_2)известно, что для запроса q запрос d_1 должен стоять раньше, чем d_2
6. R - множество всех троек, для которых известен такой порядок (обучающая выборка)
7. Строим модель $a(q, d)$, которая правильно упорядочивает документы
   a. $a(d, q_1) > a(d, q_2)$
8. Важен порядок, а не абсолютные значения

Метрики качества ранжирования

Целевая переменная

1. Упрощаем постановку задачу
2. Ответы - числа y_i (показатель релевантности)
3. Выдаем не вектор, а просто числа, по которым потом сортируем

DSG (Discount cumulative gain)

$$DCG@k(q) = \sum_{i=1}^{k} \frac{2^{y_i} - 1}{log(i+1)} \to max$$ - берем первые k документов

y_i - истинная релевантность для документа на i-й позиции

Недифференцируемая метрика

Методы ранжирования

Поточечный (pointwise) метод

1. Строим модель так, чтобы она как можно точнее приближала ответы y_i
2. $\sum_{(q,d,y)\in R} (< w, x(q, d) > - y_i)^2 \to min$
    a. где x - признаки пары запрос-документ
3. Можно использовать любые модели
4. Но: Находит точные значения y, хотя важен только порядок

Попарный подход

Прижимаем индикатор гладкой функцией, которую можно оптимизировать

Признаки в задачах ранжирования

1. Запросные
    a. Популярность запроса
    b. Тип запроса
2. Статические признаки
    a. Популярность документа
    b. Тематика
    c. Распределение слов
3. Динамические
    a. Расстояние между запросом и документов
    b. Косинусное расстояние или BM25

PageRank

1. Документы в сети ссылаются друг на друга
2. Если документа A ссылается на документ B, то он за него голосует
3. Чем меньше голосов, отдает A, тем больше вес его голоса
4. Документ B важен, если за него отдано много сильных голосов