Модель машинного обучения

На прошлом уроке мы определили функцию потерь. Зафиксируем набор объектов (*обучающую выборку*). Неформально говоря, функция потерь на вход принимает алгоритм, а на выход выдает число. Это число описывает, насколько хороши предсказания этого конкретного алгоритма на обучающей выборке. Если число большое – алгоритм предсказывает плохо, если маленькое – алгоритм предсказывает хорошо. То есть функция потерь позволяет сравнивать алгоритмы – определять, у какого алгоритма предсказания лучше.

Ясно, что мы хотим найти алгоритм, на котором функция потерь даёт как можно меньшее число – то есть достигает своего минимума. Другими словами, это алгоритм, предсказания которого лучше, чем предсказания всех других алгоритмов.

Accuracy. Если мы используем точность (accuracy), то нам хочется, чтобы точность была не минимальной, а максимальной. То есть среди всех алгоритмов мы будем искать алгоритм с максимальной точностью.

Как правило, при решении задачи машинного обучения выбирается некоторый класс алгоритмов, где каждый конкретный алгоритм из класса задаётся *параметрами* или, иначе говоря, *весами*. Зафиксировав веса, мы получим фиксированный алгоритм, его еще называют *моделью*. Тем самым, поиск наилучшего алгоритма превращается в поиск наилучших параметров. Выбирается класс моделей обычно человеком, после чего наилучшие параметры модели находятся методами математической оптимизации и зависят от данных.

Например, в этом уроке мы познакомимся с *линейной регрессией* — это класс алгоритмов (моделей), параметрами которого являются коэффициенты линейной функции.

Комментарий 1. В мета-обучении и класс моделей выбирает машина, но, к сожалению, в этой области успехов пока не так много.

Комментарий 2. В теории мы могли бы закодировать все алгоритмы специальным образом (это, кстати, можно сделать — привет машинам Тьюринга!) и дальше искать оптимальный алгоритм на всём множестве алгоритмов, не ограничиваясь конкретным классом алгоритмов. На практике к такому подходу человечество пока даже не приблизилось, поэтому решение всё-таки ищут в каком-то заранее выбранном классе алгоритмов.

Объекты превратились в константы, а алгоритмы в переменные

Концептуально только что произошло довольно важное изменение. Во всём рассуждении до этого мы считали, что признаки объекта — это переменные и они подаются на вход некоторому фиксированному алгоритму. Теперь же мы делаем ровно обратное: фиксируем некоторый набор объектов (который называем обучающей выборкой), после чего он становится набором констант. При этом алгоритм по смыслу как раз переменной. Класс алгоритмов можно представлять себе как функцию от параметров: для фиксированных параметров мы получаем конкретный алгоритм — модель, а при поиске наилучшего алгоритма параметры становятся переменными. Пояснение последней мысли можно найти в закреплённом комментарии, она может быть несколько сложной, так что не слишком задерживайтесь на ней.

Ниже — наш обучающий датасет:

Источник: гитхаб датасета

Допустим, мы ищем лучшую модель в следующем простом классе моделей. Наш класс моделей состоит из алгоритмов, которые выдают всегда один и тот же ответ — c, то есть в классе константных функций. В качестве параметра модели из этого класса можно выбрать ту самую константу, которую возвращает алгоритм.

Напомним, метка 0 соответствует пёсику, а 1 – кексику. Вот два примера алгоритмов из класса константных функций:

Пример 1. Возьмём алгоритм, который всегда возвращает 1. На этом датасете такой алгоритм имеет точность 40% – на первой и третьей картинке он даёт правильный ответ, а на второй, четвёртой и пятой – неправильный.

Пример 2. Возьмём алгоритм, который всегда возвращает 76. На этом датасете такой алгоритм имеет точность 0% – он на всех картинках даёт неправильный ответ (потому что правильные ответы это только 0 и 1).

Задача. Среди всех алгоритмов в классе константных функций найдите алгоритм, который имеет наибольшую точность (accuracy) на нашем датасете. Какова точность этого алгоритма?

Запишите ответ в процентах (например "43", без знака "%").

Введите численный ответ

Введите число

Пример

Пороговая функция как модель машинного обучения

Дата саентист Васечкин ищет работу. До этого он успел поработать в двух компаниях, а суммарно за свою жизнь побывал на 7 собеседованиях. Мы планируем сделать ему оффер, но чтобы понять, какой именно будет оптимален, хотим промоделировать принятие решений Васечкиным. Если чуть точнее, то построить бинарный классификатор, который по офферу будет определять, примет его Васечкин или нет.

Дата саентисты — люди нехитрые, вакансии они отбирают по следующему принципу: если в оффере зарплата меньше, чем 300к/сек, то отказываются, в противном случае — соглашаются. Следуя этой мудрости, будем искать решение в классе *пороговых функций*. А именно, наше решение будет иметь вид: если зарплата меньше, чем t (от слова threshold) , то Васечкин отказывается, в противном случае соглашается. Поскольку Васечкин только начинающий специалист, мы надеемся, что его устроит зарплата поменьше, из этих же соображений порог будем считать не в к/сек, а в к/мес.

Таким образом, t — это параметр модели. Предлагаемую в оффере зарплату мы обозначим за s (от слова salary). Если считать, что отказ Васечкина — это 0, согласие — 1, то наше предсказание, согласится ли Васечкин, это $\hat{f}_t(c, d, p, s) = \mathbf{1}\{s \geq t\}$. Где c, d, p, s — это признаки оффера (s мы уже определили, а c, d, p определим на следующем шаге). Другими словами, $\hat{f}_t(c, d, p, s) = 1$, если $s \geq t$, и $\hat{f}_t(c, d, p, s) = 0$, если $s < t$.

Пример. Если $t = 250$, то для любых c, d, p выполнено:

$$\hat{f}_{250}(c, d, p, 120) = 0,$$

$$\hat{f}_{250}(c, d, p, 200) = 0,$$

$$\hat{f}_{250}(c, d, p, 250) = 1,$$

$$\hat{f}_{250}(c, d, p, 300) = 1.$$

То есть равенство $t = 250$ означает, что наша модель предсказывает согласие Васечкина на любую работу с зарплатой большей или равной 250. И предсказывает отказ Васечкина от любой работы с зарплатой меньшей 250.

Комментарий: в обозначении $\hat{f}_t(c, d, p, s)$ крышечка над f обозначает то, что мы строим некоторое приближение функции f, где f это истинная функция решений Васечкина. Заметим, что f нам неизвестна. Кроме того, она может не лежать в классе пороговых функций. Нотацию с крышечкой любят использовать в математической статистике. Нижний индекс t показывает, что мы рассматриваем семейство функций с параметром t. Запись $\hat{f}(t, c, d, p, s)$ также была бы корректной, но мы предпочитаем другую, чтобы разделять параметры модели и признаки.

Вот история собеседований Васечкина:

c (компания)	d (расстояние офиса от дома Васечкина)	p (личный кабинет)	s (зарплата)	y (ответ Васечкина)
Mail.Ru	10 км	да	100 тысяч	нет
Яндекс	5 км	нет	90 тысяч	да
Пятёрочка	100 м	нет	60 тысяч	нет
Facebook	9000 км	нет	290 тысяч	нет
IBM	9055 км	нет	250 тысяч	нет
Google	9340 км	нет	350 тысяч	да
Tesla	8790 км	нет	305 тысяч	нет

Задача. Какую максимальную точность (accuracy) можно достичь в классе пороговых функций от зарплаты s?

Другим словами, найдите такое t, что $\hat{f}_t$ имеет максимальную точность, и запишите эту точность в поле ответа.

Пример. Рассмотрим $t = 120$. Это значит, что наша модель $\hat{f}_t$ предсказывает такое поведение Васечкина: если зарплата меньше 120 тысяч, то отказ, если больше ли равна 120, то согласие. Видно, что для офферов Mail.ru, Пятёрочки и Google эта модель даёт верные предсказания. А для офферов Яндекса, Facebook, IBM и Tesla – неверные. Итого на 7 объектах модель дала 3 правильных ответа и 4 неправильных (напомним, объекты это офферы). То есть точность модели равна 42.8%. Ясно, что для другого выбора t точность модели может оказаться другой.

Ответ запишите в процентах, округлив до одного знака после запятой (например "43.7", без знака "%").

Введите численный ответ

Введите число

Регрессия: оценка стоимости квартир

Линейная регрессия

Задачу регрессии можно решать, например, при помощи *линейной регрессии*. Её идея ушла не сильно дальше предыдущих примеров, состоит она в следующем: мы представили объекты в виде набора признаков, каждый из которых является некоторым числом. Давайте попробуем найти для каждого признака коэффициент такой, чтобы при сложении признаков, умноженных на эти коэффициенты, мы получали что-то близкое к нашей целевой функции.

Вспомним наши обозначения:

- r – число комнат, целое положительное число,
- d – расстояние до центра, произвольное положительное число,
- p – разрешены ли в квартире питомцы (по смыслу возможные значения «да» и «нет», но мы можем договориться, что 1 – это «да», а 0 – это «нет»).

Будем предсказывать стоимость квартиры как

$\hat{f}(r, d, p) = w_r r + w_d d + w_p p + w_0$, где коэффициенты w_r, w_d, w_p, w_0 – параметры модели, их-то мы и будем оптимизировать. Коэффициент w_0 называют *свободным коэффициентом* или *сдвигом* (bias), подробнее про него можно прочитать в закреплённом комментарии. При этом r, d, p – уже не аргументы, а константы (для каждого из объектов нашей обучающей выборки).

Пример

Линейная регрессия для задачи оценки стоимости квартир

Допустим, наша обучающая выборка состоит из 5 объектов:

r (число комнат)	d (расстояние до центра, км)	p (разрешены ли питомцы)	y (арендная плата)
2	10	1	40 тысяч
1	2	0	45 тысяч
3	15	1	60 тысяч
5	5	0	100 тысяч
2	7	0	50 тысяч

На предыдущем шаге мы решили приближать стоимость арендной платы *линейными функциями* от признаков квартир, то есть функциями вида $\hat{f}(r, d, p) = w_r r + w_d d + w_p p + w_0$, где w_r, w_d, w_p, w_0 – параметры модели.

Тогда для первой квартиры наша модель предскажет стоимость аренды $2w_r + 10w_d + w_p + w_0$ тысяч рублей в месяц. Настоящая стоимость этой аренды этой квартиры равна 40 тысяч. Поэтому квадратичная функция потерь для первой квартиры будет равняться $(40 - (2w_r + 10w_d + w_p + w_0))^2$. А для всей обучающей выборки функция потерь будет
равняться среднему значению квадратичной функции потерь для всех квартир из выборки, то есть:

$$L(w_r,\ w_d, w_p, w_0) = \frac{1}{5}((40 - (2w_r + 10w_d + w_p + w_0))^2 + (45 - (w_r + 2w_d + w_0))^2 + \ (60 - (3w_r + 15w_d + w_p + w_0))^2 + \ (100 - (5w_r + 5w_d + w_0))^2 + \ (50 - (2w_r + 7w_d + w_0))^2).$$

Если бы в нашей таблице было не 5 квартир, а 5 миллионов, а вместо 3 признаков — 300, содержательно ничего бы не изменилось, только выражение для функции потерь стало бы ещё более громоздким.

Выводы: обучение — это поиск минимума

По своему смыслу функция потерь отражает, насколько далеки предсказания нашей модели от правильных ответов на обучающем датасете. Таким образом, задача поиска лучшей модели в некотором классе моделей сводится к поиску параметров (весов) модели, при которых значение функции потерь минимально на обучающем датасете. В предыдущем примере мы свели задачу к поиску значений параметров w_r, w_d, w_p и w_0, минимизирующих значение функции потерь $L(w_r,\ w_d, w_p, w_0)$ — конкретной функции от переменных w_r, w_d, w_p и w_0 для фиксированного обучающего датасета. Наш пример игрушечный, тем не менее утверждение «обучение = поиск точки минимума» по большому счёту верно.

Для функции, которую мы выписали на предыдущем шаге, найти минимум "голыми руками" уже гораздо сложнее, чем в первых двух примерах. В первой части программы Математика для Data Science мы рассказываем про наиболее общий подход к решению задачи поиска минимума — *градиентный спуск*. Чтобы ловко обращаться с *градиентом,* мы подробно изучим такие важные математические понятия как предел, производная, дифференциал и многие другие.

В этих уроках не было совсем строгих рассуждений, однако, надеемся, что вы увидели, почему строгий формальный подход может быть полезен. Такому подходу мы постараемся вас научить в нашей программе.

Ура, вот вы и сделали первые шаги в мир математики для Data Science!

А вот и задачка на десерт:

Источник: Carolyn Bratnober на queerbiblecomics