

Введение в машинное обучение

Национальный исследовательский университет "Высшая школа экономики"
Yandex School of Data Analysis

Неофициальный конспект по курсу.

1 апреля 2021 г.

1 Проблема переобучения

Одна из основных проблем в машинном обучении — проблема переобучения. Рассмотрим пару примеров:

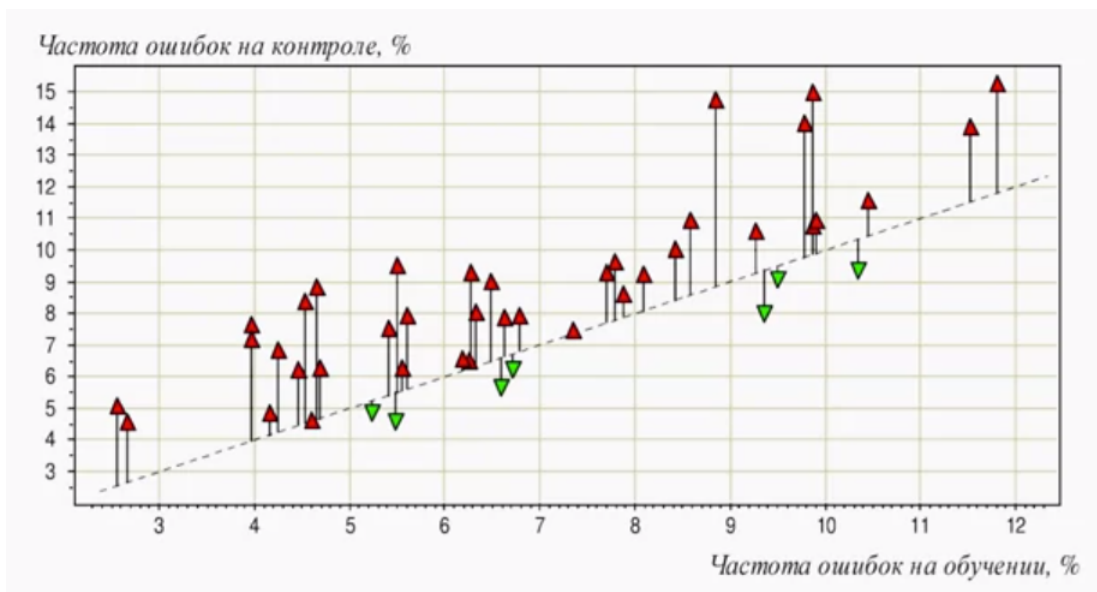
1.1 Пример. Переобучение в задачи медицинской диагностики.

Задача предсказания отдаленного результата хирургического лечения атеросклероза.

Задача в целом типичная. Данные в ней содержат разнотипные признаки с пропусками, причем признаки делятся на 2 большие группы: данные о гемодинамике (скорость кровотоков в артериях, венах), данные об иммунологии.

Задача состоит в том, чтобы предсказать, будут ли у человека осложнения после хирургического лечения атеросклероза, когда человеку кусок вены заменяют шунтом. Это важно для хирургов, чтобы как можно раньше заметить различные осложнения.

И при решении этой задачи (у лектора) возникли проблемы переобучения:



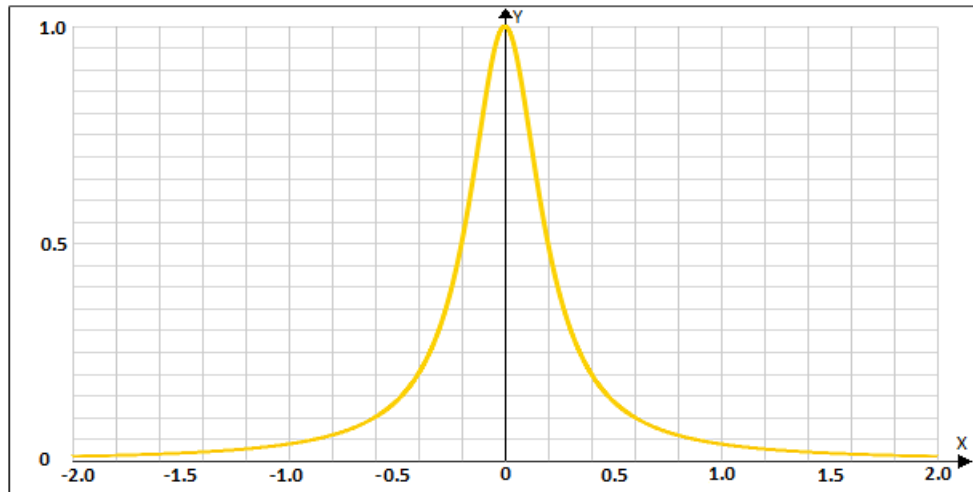
Точки на данном графике — это отдельные предсказательные модели, отдельные алгоритмы. По осям — частота ошибок на обучении и частота ошибок на контроле. Пунктирная линия — это биссектриса (проходит под углом 45 градусов), она показывает геометрическое место точек, где частота ошибок на контроле и обучении совпадает.

На графике видно, что все что множество точек имеет систематический сдвиг вверх, то есть в среднем частота ошибок на контроле больше. То есть алгоритм на практике становится не таким точным, как хотелось бы — мы переобучили до соответствия обучающей выборке.

1.2 Пример. Переобучение полиномиальной регрессии

Второй пример уже из области регрессии (пример искусственный, но все же).

Возьмем функцию $y(x) = \frac{1}{1+25x^2}$ на отрезке $x \in [-2, 2]$ и попробуем ее аппроксимировать.



А в качестве признаков описаний объектов будем использовать векторы степеней x :

$$x \mapsto (1, x^1, x^2, \dots, x^n)$$

В таком случае можно попробовать использовать линейную модель регрессии, а именно — модель полиномиальной регрессии (у нас получится полином, потому что полиномиальная):

$$a(x, \theta) = \theta_0 + \theta_1 x + \dots + \theta_n x^n \quad \text{— полином степени } n.$$

Совершенно стандартный подход к решению этой задачи — метод наименьших квадратов, то есть, на языке машинного обучения, использование квадратичных функций потерь:

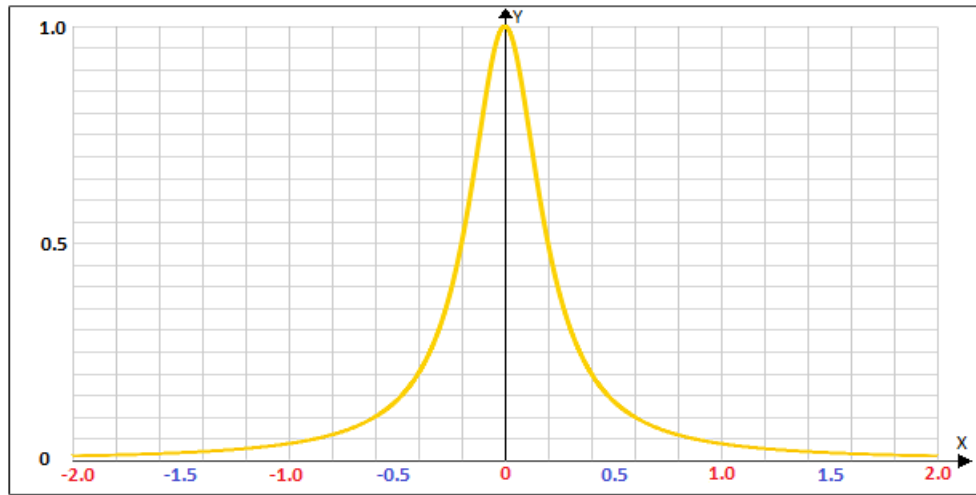
$$Q(a, X^l) = \sum_{i=1}^l (\theta_0 + \theta_1 x_i + \dots + \theta_n x_i^n - y_i)^2 \rightarrow \theta_0, \dots, \theta_n \underset{\min}{}$$

Далее возьмем обучающую и контрольную выборки следующим образом:

$$X^l = \{x_i = 4 \frac{i-1}{l-1} - 2 \mid i = 1, \dots, l\}$$

$$X^k = \{x_i = 4 \frac{i-0.5}{l-1} - 2 \mid i = 1, \dots, l-1\}$$

Выглядит странно, но на деле это нам даст обучающую выборку в узлах сетки, а контрольная выборка будет содержать точки ровно посередине между двумя соседними обучающими точками:



И наконец зададимся вопросом, что происходит с функционалом качества на обучающей выборке и контрольной выборке.

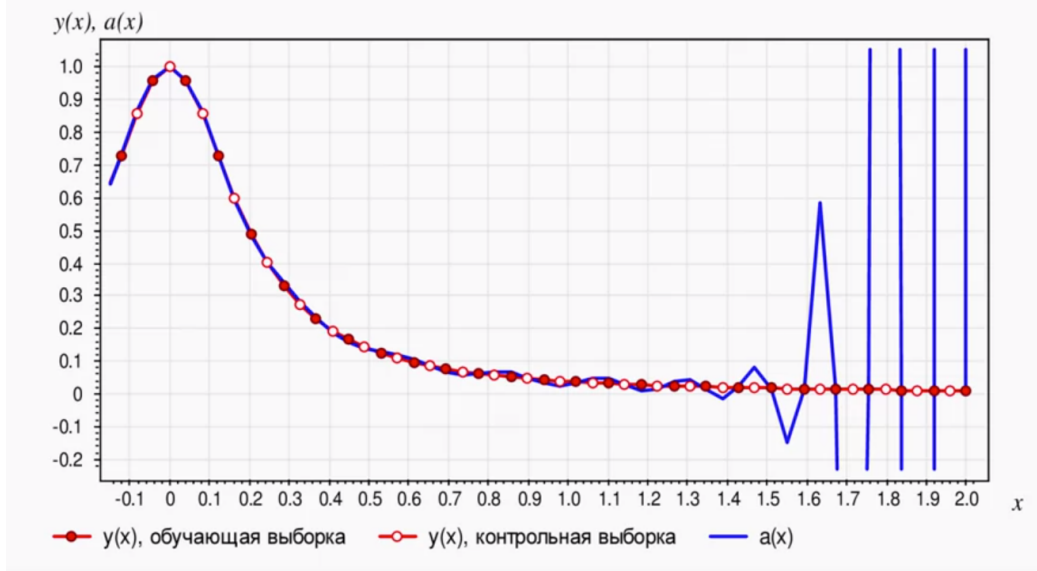
Нарисуем их как две зависимости среднеквадратичной ошибки от степени полинома ($l = 50$, $n = 1, \dots, n_{31}$):



Можно увидеть, что ошибка на обучении монотонно убывает (чем больше степень полинома, тем больше коэффициентов степеней свободы, тем точнее соответствие функции), но в какой-то момент степень полинома начинает нам "мешать и качество восстановления зависимости на контрольной выборке, начиная с 21 степени начинает только ухудшаться.

Вопрос: почему и что происходит при этом с самой функцией?

$$y(x) = \frac{1}{1 + 25x^2}; \quad a(x) \text{ — полином степени } n = 38$$



На данном графике **красным** изображена зависимость, которую мы аппроксимируем (исходная функция), а **синим** — то, чем мы ее аппроксимируем (наш полином степени 38 в данном случае).

Тут же мы видим, что на концах отрезка полином очень плохо описывает эту функцию, хотя очень точно проходит через точки обучающей выборки. Между точками обучающей выборки полином "улетает" куда-то ближе к бесконечности.

Данный эффект связан с тем, что при увеличении числа степеней свободы (числа параметров модели) у нас начинает происходить подстройка под конкретные точки выборки, и модель ловит несоответствия между моделью и оригинальной функцией. Такая модель может быть очень неустойчива к шумам (погрешностям измерений, неточным данным) — значит нужно как-то ограничивать сложность модели.

Как избежать этого эффекта? Как научиться оценивать этот эффект? Как строить такие модели, где гарантировано степень переобучения будет в некоторых пределах, например, 2%?

1.3 Эмпирические оценки обобщающей способности

Для того, чтобы измерять подобные эффекты, принято использовать функционалы, которые непосредственно измеряют качество построенной модели на тех данных, на которых модель не обучалась.

- Эмпирический риск на тестовых данных (**hold-out**):

$$HO(\mu, X^l, X^k) = Q(\mu(X^l), X^k) \rightarrow \min$$

Напомним, что $\mu(X^l)$ — это минимизация некоторого функционала качества (не указан явно, но подразумевается) на некоторой выборке (X^l) .

И $Q(a, X^k)$ — функционал качества некоторого алгоритма a на некоторой выборке X^k .

В данном случае *разделение выборок* происходит самым простым способом — выборка между обучением и контролем в некотором соотношении: X^l и X^k соответственно.

Однако, есть проблема, когда при *неудачном выборе разбиения* наша оценка будет *субъективной, смещенной*. Чтобы этого не происходило, используется метод, описанный далее.

- Скользящий контроль (**leave-one-out**), $L = l + 1$:

$$LOO(\mu, X^L) = \frac{1}{L} \sum_{i=1}^L \mathcal{L}(\mu(X^L \setminus \{x_i\}), x_i) \rightarrow \min$$

Так же напомним, что $\mathcal{L}(a, x)$ — функция потерь — величина ошибки алгоритма a на объекте x .

Решение проблемы «*неудачного разбиения*» здесь решается тем, что делается *много* разбиений.

Простейший способ это сделать — каждый раз выделять по *одному контрольному объекту*, а по оставшейся выборке проводить обучение. *Минус* в том, что обучение придется проводить столько раз, сколько объектов в обучающей выборке (может тысяча, а может миллион... долго в общем).

- Кросс-проверка (**cross-validation**) по N разбиениям, $X^L = X_n^l \sqcup X_n^k$, $L = l + k$:

$$CV(\mu, X^L) = \frac{1}{N} \sum_{n=1}^N Q(\mu(x_n^l), X_n^k) \rightarrow \min$$

Метод делает всевозможные кросс-проверки (как следует из названия), когда мы каким-то образом устраиваем много N разбиений выборки на обучение и контроль, и каждый раз мы на одной части обучаемся, на другой проверяем результат, и в итоге усредняем результат по всевозможным разбиениям.

Данный способ уже менее зависим от того, какие разбиения будут выбраны. Можно в том числе делать случайные разбиения.

2 Эксперименты в машинном обучении

2.1 Эксперименты на реальных данных

1. Эксперименты на конкретной прикладной задаче:

- цель — решить задачу как можно лучше
- важно понимание задачи и данных
- важно придумывать информативные признаки
- конкурсы по анализу данных: <https://www.kaggle.com>

2. Эксперименты на наборах прикладных задач:

- цель — протестировать метод в разнообразных условиях
- нет необходимости (и времени) разбираться в сути задач (:с)
- признаки, как правило, уже кем-то придуманы
- репозитории UC Irvine Machine Learning Repository <https://archive.ics.uci.edu/ml> (308 задач, 09-02-2015)

Часто берется большой массив задач с хорошо (понадемся на это) подготовленными данными, и по этим задачам прогоняется большое число алгоритмов. Обычно строится матрица «задача-методы», и в каждой ячейке этой матрицы находятся те самые оценки **CV** или **LOO**, и показывается наличия и степени переобучения различных методов.

По этим данным можно уже разрабатывать, подгонять алгоритм, чтобы он был как можно более универсален или по крайней мере не имел серьезных проблем с переобучением.

Этот тип экспериментов скорее нужен для исследователей, которые занимаются разработкой новых методов, а не решением каких-то задач.

2.2 Эксперименты на модельных (синтетических) данных

Используются для тестирования новых методов обучения.

Преимущество — мы знаем истинную $y(x)$ (**ground truth**) т.к. мы на ее основе генерируем данные. Более того, поскольку мы выбираем нашу зависимость $y(x)$, мы можем устроить целую линейку задач от *очень простой* до *очень сложной*, и посмотреть, где алгоритм уже не справляется.

1. Эксперименты на модельных (syntetic) данных:

- цель — отладить метод, выявить границы применимости
- объекты x_i из придуманного распределения (часто 2D)
- ответы $y_i = y(x_i)$ для придуманной функции $y(x)$
- двумерные данные + визуализация сборки

2. Эксперименты на полумодельных (semi-syntetic) данных:

- цель — протестировать помехоустойчивость модели
- объекты x_i из реальной задачи (+ шум)
- зависимость y_i все еще искусственная (выбирается на реальных данных)
- ответы $y_i = a(x_i)$ для полученного решения $a(x)$ (+ шум)

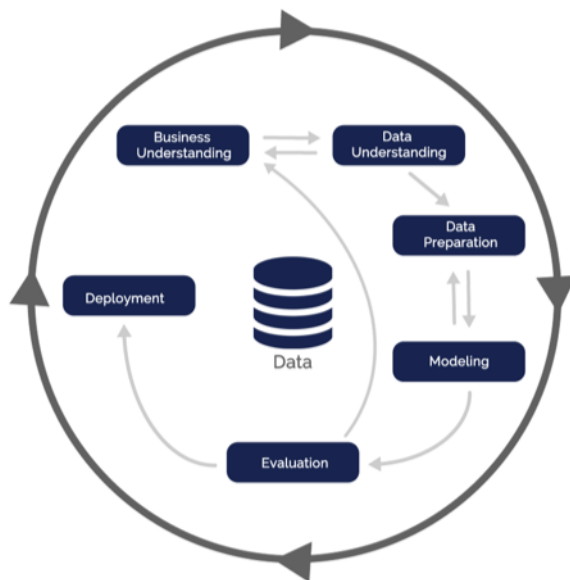
3 CRISP-DM: Cross Industry Standard Process for Data Mining

CRISP-DM — **C**ross **I**ndustry **S**tantard **P**rocess for **D**ata **M**ining

(да, я просто выделил буквы, если кому было лень читать название, чтобы уж так было понятнее)

Говоря о решении задач в реальных приложениях (в науке, бизнесе, медицине...) можно вспомнить про межотраслевые стандарты. Одним из них является **CRISP-DM** — межотраслевой стандарт решения задач интеллектуального анализа данных.

CRISP-DM содержит в себе **формализацию схемы решения прикладных задач на реальных данных**, которую, в общем-то, сообщество выработало уже давно. Он предполагает решение задачи в 5 шагов, которые могут при этом замыкаться и повторяться многократно:



1. Понять, откуда задача пришла (понять предметную область, сферу).
2. Понять, как собирались данные: если ли там шумы, пропуски, выбросы, все ли признаки полезны.
3. Выделить полезные данные по сырой информации — *подготовка данных*
4. Моделирование — построение предсказательной модели (основное).
5. Оценка полученных моделей.
6. Внедрение модели в производственные процессы (окончание задачи).

4 Резюме

Этапы решения задач машинного обучения

- понимание задачи и данных;
- предобработка данных и изобретение признаков;
- построение модели;
- сведение обучения к оптимизации;
- решение проблем оптимизации и переобучения;
- оценивание качества решения;
- внедрение и эксплуатация.

Красным выделены основные моменты, которыми мы будем заниматься.