Урок 3

Проблема переобучения и борьба с ней

3.1. Проблема переобучения

3.1.1. Пример: проблема переобучения в задачах классификации

Допустим при решении задачи классификации был построен некоторый алгоритм, например линейный классификатор, причем доля ошибок на объектах из обучающей выборки была равна 0.2, и такая доля ошибок является допустимой.

Но поскольку алгоритм не обладает обобщающей способностью, нет никаких гарантий, что такая же доля ошибок будет для новой выборки. Вполне может возникнуть ситуация, что для новой выборки ошибка станет равной 0.9. Это значит, что алгоритм не смог обобщить обучающую выборку, не смог извлечь из нее закономерности и применить их для классификации новых объектов. При этом алгоритм как-то смог подогнаться под обучающую выборку и показал хорошие результаты при обучении без извлечения истинной закономерности. В этом и состоит проблема переобучения.

3.1.2. Пример: проблема переобучения в задачах линейной регрессии

Глубже понять проблему переобучения можно на данном примере. На следующем графике изображена истинная зависимость и объекты обучающей выборки:

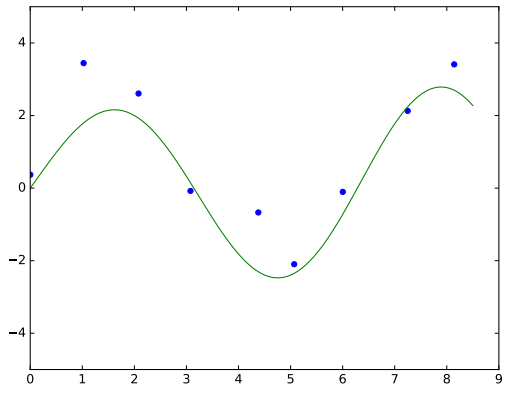

Рис. 3.1: Истинная зависимость (зеленая линия) и элементы обучающей выборки (изображены синими точками).

Видно, что истинная зависимость является нелинейной и имеет два экстремума.

В модели $a(x) = w_0$, после того, как она будет настроена под данные, на графике получается некоторая горизотальная кривая, которая довольно плохо обобщает информацию об объектах из выборки.

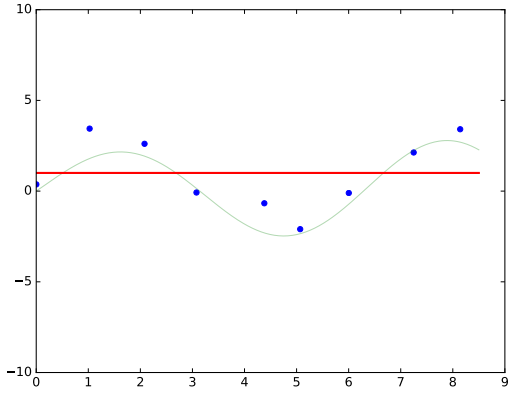

Рис. 3.2: Модель $a(x) = w_0$.

Имеет место недообучение. Хороший алгоритм не был построен, поскольку семейство алгоритмов слишком мало и с его помощью невозможно уловить закономерность.

В линейной регрессии используется семейство алгоритмов $a(x) = w_0 + w_1 x$.

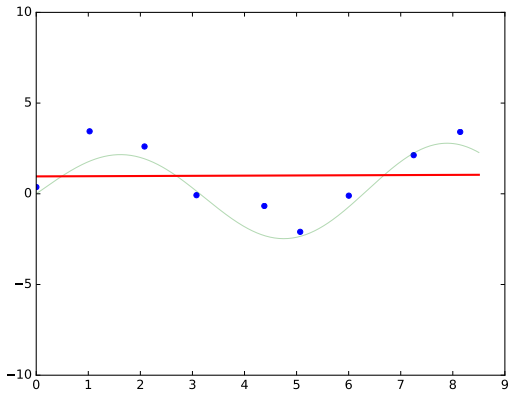

Рис. 3.3: Модель $a(x) = w_0 + w_1 x$.

В этом случае также будет иметь место недообучение. Получилось лучше, но прямая тоже плохо описывает данные.

Если семейство алгоритмов — множество многочленов 4-ей степени:

$$a(x) = w_0 + w_1 x + w_2 x^2 + ... + w_4 x^4,$$

то после обучения получившаяся кривая будет достаточно хорошо описывать и обучающую выборку, и истинную зависимость.

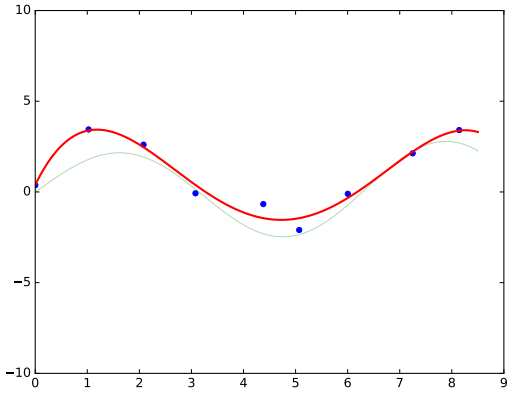

Рис. 3.4: Модель $a(x) = w_0 + w_1 x + w_2 x^2 + ... + w_4 x^4$.

В таком случае качество алгоритма хорошее, но нет идеального совпадения. Встает вопрос, а можно ли добиться совпадения увеличением сложности алгоритма.

При использовании многочленов 9-ой степени уже имеет место переобучение.

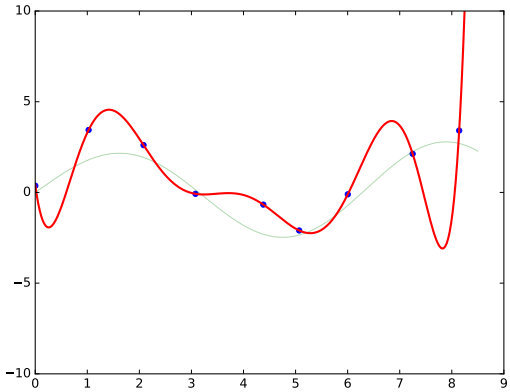

Рис. 3.5: Модель $a(x) = w_0 + w_1 x + w_2 x^2 + ... + w_9 x^9$.

Восстановленная зависимость дает идеальные ответы на всех объектах обучающей выборки, но при этом в любой другой точке сильно отличается от истинной зависимости. Такая ситуация называется переобучением. Алгоритм слишком сильно подогнался под обучающую выборку ценой того, что он будет давать плохие ответы на новых точках.

3.1.3. Недообучение и переобучение

Таким образом, недообучение — ситуация, когда алгоритм плохо описывает и обучающую выборку, и новые данные. В этом случае алгоритм необходимо усложнять.

В случае переобучения, данные из обучающей выборки будут описываться хорошо, а новые данные плохо. Выявить переобучение, используя только обучающую выборку, невозможно, поскольку и хорошо обученный, и переобученный алгоритмы будут хорошо ее описывать. Необходимо использовать дополнительные данные.

Существуют несколько подходов к выявлению переобучения:

- Отложенная выборка. Часть данных из обучающей выборки не участвуют в обучении, чтобы позже проверять на ней обученный алгоритм.
- Кросс-валидация, несколько усложненный метод отложенной выборки. (Об этом способе речь пойдет позже.)

- Использовать меры сложности модели. Об этом пойдет речь далее.

3.2. Регуляризация

В этом разделе речь пойдет о регуляризации — способе борьбы с переобучением в линейных моделях.

3.2.1. «Симптомы» переобучения. Мультиколлинеарность.

Мерой сложности, то есть «симптомом» переобученности модели, являются большие веса при признаках. Например, в предыдущем разделе при обучении модели

$$a(x) = w_0 + w_1 x + w_2 x^2 + ... + w_9 x^9$$

веса оказывались огромными:

$$a(x) = 0.5 + 12458922x + 43983740x^2 + ... + 2740x^9.$$

Другая ситуация, в которой можно встретиться с переобучением — мультиколлинеарность. Так называется проблема, при которой признаки в выборке являются линейно зависимыми. Другими словами, существуют коэффициенты $\alpha_1, ..., \alpha_d$ такие, что для любого объекта x_i из выборки выполняется:

$$\alpha_1 x_i^1 + ... + \alpha_d x_i^d = 0.$$

Более компактно последнее выражение можно переписать в виде:

$$\langle \alpha, x_i \rangle = 0.$$

Допустим, было найдено решение задачи оптимизации:

$$w_* = \text{argmin}_w \frac{1}{\ell} \sum_{i=1}^{\ell} \big(\langle w, x_i \rangle - y_i \big)^2$$

Другой вектор весов, полученный сдвигом в направлении вектора α:

$$w_1 = w_* + t\alpha,$$

так как для элементов x выборки выполняется:

$$\langle w_* + t\alpha, x \rangle = \langle w_*, x \rangle + t\langle \alpha, x \rangle = \langle w_*, x \rangle,$$

также будет являться решением задачи оптимизации. Другими словами, он будет также хорошо описывать данные в выборке, как и исходный алгоритм. Фактически, решениями задачи оптимизации являются бесконечное множество алгоритмов, но многие из них имеют большие веса, и далеко не все обладают хорошей обобщающей способностью. Поэтому здесь тоже легко столкнуться с переобучением.

3.2.2. Регуляризация

Выше было продемонстрировано, что если веса в линейной модели большие, существует высокий риск переобучения. Чтобы бороться с этим, минимизируется уже не выражение для функционала ошибки $Q(a, X)$, а новый функционал, получаемый прибавлением регуляризатора. Самый простой регуляризатор — квадратичный регуляризатор:

$$\|w\|^2 = \sum_{j=1}^{d} w_j^2.$$

В этом случае имеет место следующая задача оптимизации:

$$Q(w, X) + \lambda \|w\|^2 \to \min_w .$$

Таким образом, при обучении будет учитываться также то, что не следует слишком сильно увеличивать веса признаков.

3.2.3. Коэффициент регуляризации

Введенный выше коэффициент λ, который стоит перед регуляризатором, называется коэффициентом регуляризации. Чем больше λ, тем ниже сложность модели. Например, при очень больших его значениях оптимально просто занулить все веса. В то же время при слишком низких значениях λ высок риск переобучения, то есть модель становится слишком сложной.

Поэтому нужно найти некоторое оптимальное значение λ, достаточно большое, чтобы не допустить переобучения, и не очень большое, чтобы уловить закономерности в данных. Обычно λ подбирается на кросс-валидации, о которой пойдет речь в следующем разделе.

3.2.4. Смысл регуляризации

Чтобы понять смысл регуляризации, вместо задачи оптимизации с квадратичным оптимизатором нагляднее рассмотреть задачу условной оптимизации:

$$\begin{cases} Q(w, X) \to \min\limits_{w} \\ \|w\|^2 \leq C \end{cases}$$

Добавление регуляризатора вводит требование, чтобы решение задачи минимизации искалось в некоторой круглой области с центром в нуле.

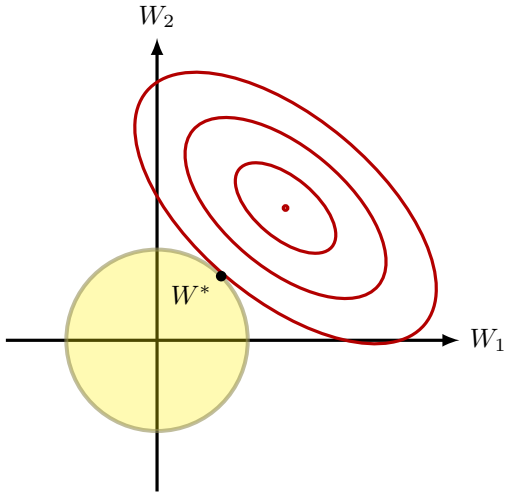

Рис. 3.6: Геометрический смысл условной регуляризации. Красная точка — настоящий оптимум функции, красные линии — линии уровня функции, черная точка — оптимум функции при введенном ограничении.

Таким образом, решение задачи с регуляризатором не будет характеризоваться слишком большими значениями весовых коэффициентов.

3.2.5. Виды регуляризаторов

Рассмотренный выше квадратичный регуляризатор (L_2-регуляризатор) является гладким и выпуклым, что позволяет использовать градиентный спуск.

Также существует L_1-регуляризатор:

$$\|w\|_1 = \sum_{j=1}^{d} |w_j|,$$

который представляет собой L_1-норму вектора весов. Он уже не является гладким, а также обладает интересным свойством. Если применять такой регуляризатор, некоторые веса оказываются равными нулю. Другими словами, такой регуляризатор производит отбор признаков и позволяет использовать в модели не все признаки, а только самые важные из них.

3.3. Оценка качества алгоритмов. Кросс-валидация

В этом разделе речь пойдет об оценке качества алгоритмов и о том, как понять, как поведет себя алгоритм на новых данных.

3.3.1. Выявление переобучения

Уже было сказано, что переобучение сложно выявить, используя только обучающую выборку: и хороший, и переобученный алгоритмы будут показывать хорошее качество на объектах обучающей выборки. Рассмотренные в предыдущем разделе меры переобученности (значения регуляризаторов), безусловно, можно применять, но они не дают ответа на вопрос, насколько хорошо алгоритм поведет себя на новых данных, то есть какая у него будет доля ошибок на новых данных.

3.3.2. Отложенная выборка

Самый простой способ оценить качество алгоритма — использование отложенной выборки. В этом случае следует разбить выборку на две части: первая из двух частей будет использоваться для обучения алгоритма, а вторая, тестовая выборка, — для оценки его качества, в том числе для нахождения доли ошибок в задаче классификации, MSE (среднеквадратичной ошибки) в задаче регрессии и других мер качества в зависимости от специфики задачи.

Естественный вопрос — о том, в какой пропорции производить разбиение. Если взять тестовую выборку слишком маленькой, оценка качества будет ненадежной, хотя обучающая выборка будет почти совпадать с полной выборкой. В противоположенном случае, если отложенная часть будет большой, оценка качества будет надежной, но низкое качество алгоритма может свидетельствовать о недостаточном объёме первой, обучающей, части выборки. Обычно выборку разбивают в соотношениях $70/30$, $80/20$ или $0.632/0.368$.

Преимуществом отложенной выборки является то, что обучать алгоритм приходится всего лишь один раз, но при этом результат сильно зависит от того, как было произведено разбиение.

Например, оценивается стоимость жилья по некоторым признакам. И есть особая категория жилья, например двухэтажные квартиры. И если окажется, что все двухэтажные квартиры, которых немного, попали в отложенную выборку, то после обучения алгоритм будет давать на них очень плохое качество, поскольку в обучающей выборке таких объектов не было.

Чтобы решить эту проблему, можно использовать следующий подход: построить n различных разбиений выборки на 2 части, для каждого разбиения найти оценку качества, а в качестве итоговой оценки качества работы алгоритма использовать усредненное по всем разбиениям значение. Но и в данном случае, поскольку разбиения строятся случайно, нет никаких гарантий, что особый объект хотя бы раз попадет на обучение.

3.3.3. Кросс-валидация

Более системный подход — кросс валидация. В этом случае выборка делится на k блоков примерно одинакового размера. Далее по очереди каждый из этих блоков используется в качестве тестового, а все остальные — в качестве обучающей выборки.

После того, как каждый блок побывает в качестве тестового, будут получены k показателей качества. В результате усреднения получается оценка качества по кросс-валидации.

При этом встает вопрос, какое число блоков использовать. Если блоков мало, получаются надежные, но смещенные оценки. В случае большого числа блоков оценки, наоборот, получаются ненадежными (большой разброс оценок), но несмещенными.

Нет конкретных рекомендаций относительно выбора k. Обычно выбирают $k = 3, 5, 10$. Чем больше k, тем больше раз приходится обучать алгоритм. Поэтому на больших выборках следует выбирать небольшие значения k, так как даже при удалении $1/3$ выборки (а она большая) оставшихся данных будет достаточно для обучения.

3.3.4. Совет: перемешивайте данные в выборке

Часто данные в файле записаны в отсортированном виде по какого-нибудь признаку. Поэтому всегда следует перемешивать выборку прежде, чем производить кросс-валидацию. В ином случае алгоритм будет показывать плохое качество и причина этого будет не так очевидна.

При этом есть задачи, в которых выборку нельзя перемешивать. Это задачи предсказания будущего, например предсказание погоды на следующий день. В этом случае нужно особо следить за тем, как происходит деление выборки.

3.4. Выбор гиперпараметров и сравнение алгоритмов

В этом разделе речь пойдет о выборе гиперпараметров и сравнении различных алгоритмов с помощью изученных ранее схем.

3.4.1. Гиперпараметры

Гиперпараметрами называются такие параметры алгоритмов, которые не могут быть получены из обучающей выборки при обучении, поэтому их надо подбирать путем многократного обучения алгоритма. Примерами гиперпараметров являются:

- Параметр регуляризации λ (при использовании регуляризатора)
- Степень полинома в задаче регрессии с семейством алгоритмов, заданным множеством полиномов определенной степени.

3.4.2. Сравнение разных алгоритмов

Более общая задача — сравнение разных алгоритмов:

- обученных с разными значениями гиперпараметров;
- использующих различный способ регуляризации;
- настроенных с использованием разного функционала ошибки, например среднеквадратичной ошибки и средней абсолютной ошибки;
- которые принадлежат разным классам алгоритмов.

При сравнении алгоритмов можно использовать как отложенную выборку, так и кросс-валидацию, но при этом следует соблюдать осторожность.

Действительно, пусть 1000 алгоритмов сравниваются по качеству на отложенной выборке. Каждый из 1000 алгоритмов, обученных на обучающей выборке, тестируется на отложенной, и в результате выбирается лучший. Фактически на этом шаге отложенная выборка также становится своего рода обучающей, и возникает проблема переобучения: из большого числа алгоритмов выбирается тот, который лучше всего ведет себя на отложенной выборке, лучше подогнан под нее.

3.4.3. Улучшенная схема сравнения алгоритмов

Чтобы бороться с этим, следует использовать несколько усовершенствованную схему оценивания качества алгоритмов, а именно все данные нужно будет делить на 3 части (в случае использования отложенной выборки): обучение, валидация и контроль. Каждый из тысячи алгоритмов будет обучен на обучающей выборке, а его качество будет измерено на валидационной. Алгоритм с наилучшим качеством будет проверен на тестовой выборке, чтобы исключить переобучение и проверить алгоритм на адекватность. По сути именно тестовая выборка будет играть роль новых данных.

Если предпочтительно использовать кросс-валидацию, то данные следует разбить на 2 части. Первая из них будет использоваться для обучения алгоритмов и оценки качества с помощью кросс-валидации, после чего лучший алгоритм будет проверен на адекватность на контрольной выборке.